THE BEST COMPUTER PROJECTS MAGAZINE MARCH 1984

— WIN A — BROTHER EP44

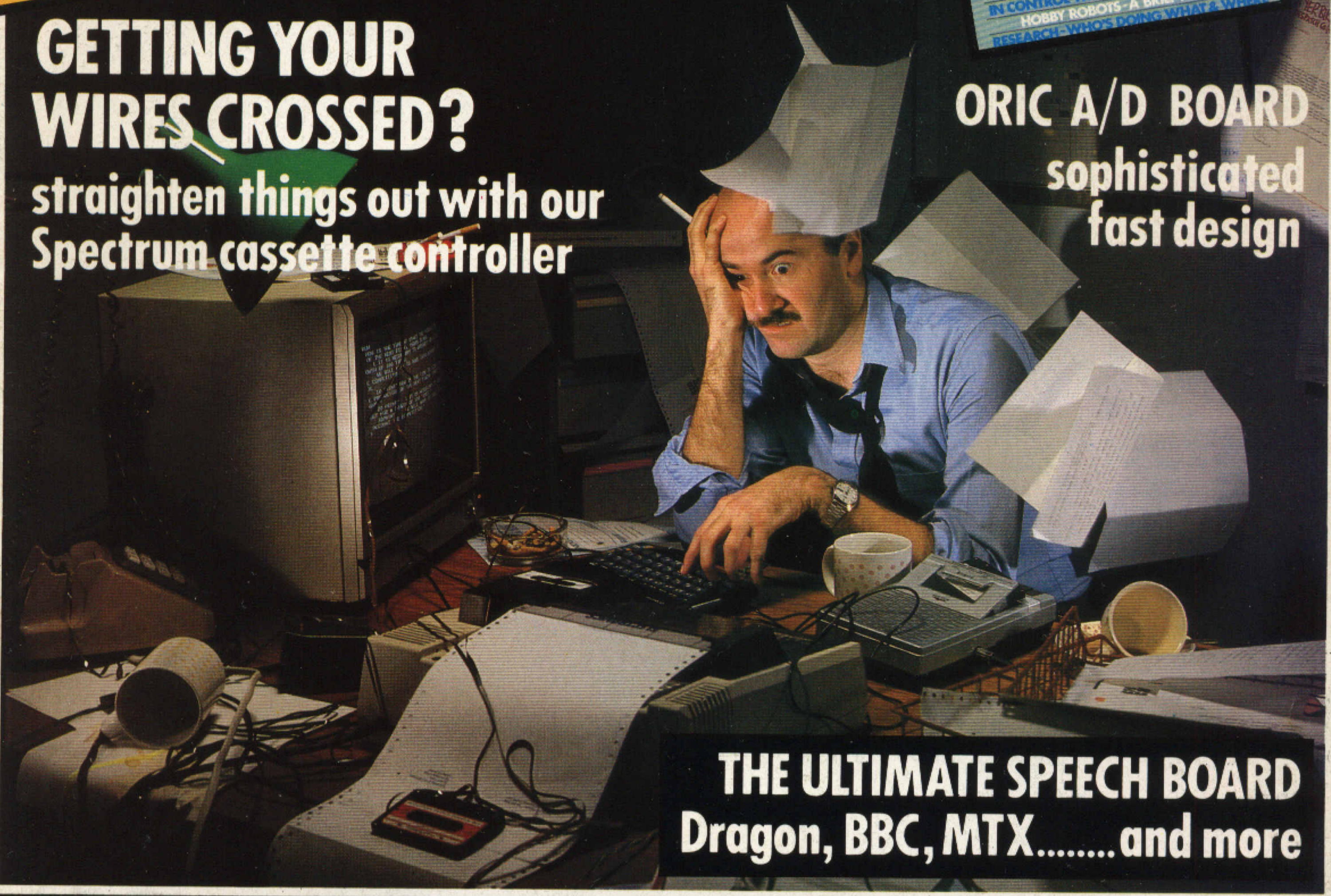
ELECTRONICS & COMPUTING MONTHLY

6 Books 18 38 53 Kits 55 PC Board 66 69 6502 Data
8 Oric 24 42 54 Kits 70 Books
34 56
14 Oric

AN EMAP PUBLICATION
USA $2.95
Germany D5.80
85p

FREE FREE

A new magazine for you and your robot



GETTING YOUR WIRES CROSSED?

straighten things out with our Spectrum cassette controller

ORIC A/D BOARD

sophisticated fast design

THE ULTIMATE SPEECH BOARD

Dragon, BBC, MTX........and more

SINCLAIR QL-DEATH OF THE 8-BIT MICRO

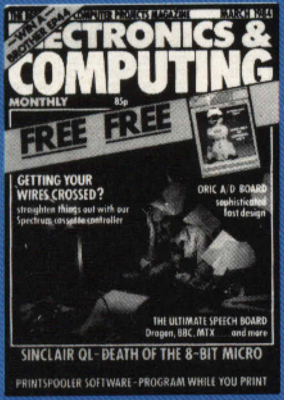
BBC 2ND PROCESSORS-WHO DARES WINS

PRINTSPOOLER SOFTWARE-PROGRAM WHILE YOU PRINT

ELECTRONICS & COMPUTING

Contents

VOLUME 5 ISSUE 3



Electronics & Computing Monthly
Scriptor Court, 155 Farringdon Road,
London, EC1R 3AD

Editorial 01-833-0846
Editor Gary Evans
Assistant Editor William Owen
Administration Liz Gregory

Advertising 01-833-0531
Manager Claire Fullerton
Executive Richard Jansz
Classified Andrew King

Production 01-833-0531
Art Editor Jeremy Webb
Make-up Time Graphics
Publisher Alfred Rolington

Distribution
EMAP National Publications

Published by
EMAP Business and Computer Publications

Printed by
Riverside Press, England

Subscriptions
Electronics & Computing Monthly, (Subscription Department), Competition House, Farndon Road, Market Harborough, Leicestershire.

Electronics & Computing Monthly is normally published on the 13th day of each month

PROJECTS

FEATURES

REVIEWS

PLUS

The second part of our multiprocessing feature and the fourth instalment of the SAT 16 project have been held over due to lack of space.

NEWS NEWS NEWS NEWS

Atmos: a better keyboard, a better ROM.

Better late than never?

The Atmos **is** a little more than an Oric 1 under another name. Oric's latest machine has a vastly improved keyboard; a new ROM operating system which, claim the makers, greatly enhances performance; and several new commands.

Some 160,000 Oric 1s were sold in 1983, but the machine was plagued by technical problems, unreliability, and in its earliest days by delivery problems. The machine was never able to compete on equal terms with the Spectrum, and in particular lacked comparable software support (according to Sir Clive, there are now some 5000 different commercial packages available for the Spectrum).

The Atmos is intended to make up for the Oric 1s defficiencies. For £170 (£40 more than the Spectrum) the new machine offers three major benefits over its rival. Firstly, the aforementioned keyboard with full-travel typewriter style keys; secondly (and no change here) a centronics interface enabling connection to a wide range of printers; and thirdly, RGB output. Cassette tape loading has also been improved, with a new facility to override error checking (overcoming errors in the header alone) and verification, storage and recall of arrays.

New commands include GRAB, allowing better utilisation of available memory, and the bug in the Microsoft FRE command has been eradicated. All existing Oric 1 programs will be compatible with the new machine. The Oric printer and disc drives will also be available in the same attractive packaging as the Atmos.

The Atmos is clearly a considerable improvement over its forebear, but the manufacturers may have missed the bus – the question mark over software support remains.

I triple E for the BBC

Acorn Computers have developed an IEEE general purpose interface bus for the BBC micro. Called the Acorn IEEE Interface (Acorn do not have the dash of Sinclair when it comes to original names) the device connects directly to the BBC micro to provide computer control over a network of up to 14 separate IEEE-488 compatible devices, including 'scopes, voltmeters, logic analysers, spectrum analysers, etc.

The new interface will further establish the BBC machine as a scientific, laboratory, and development tool. Together the interface and computer should provide a low cost and versatile control system, and coupled with the high level control languages now available on the BBC, sophisticated analysis programs would be able to directly process data collected through the new interface.

The Acorn IEEE consists of a self-contained unit and a ROM chip which plugs into the sideways ROM board of the BBC. The ROM provides the IEEE filing system, a set of plain English commands which the programmer uses to pass instructions from the micro to the equipment on the network.

At £325, the interface costs nearly as much as the computer itself. By industrial standards, this is not so expensive and educational and scientific establishments will not be deterred – hobbyists may be. *E&CM* is hoping to publish a construction project for the BBC IEEE interface in the near future, and this will cost around £80-£100 – a figure which is a little nearer the mark.

Meanwhile CST (Cambridge Systems Technology) has 'admitted failure' in its attempt to manage sales of their Procyon IEEE interface for the BBC. The Procyon was launched on the market in September '83, and CST have found it impossible to meet what they call a 'staggering demand'.

A waiting game

Clive Sinclair and his team have once again assured themselves of a healthy share of the profitable PC market with the launch of the QL. Like its forerunners, the ZXs 80, 81, and Spectrum, the QL has managed to break the mould of British computers. It has done this by offering a price/performance ratio that is, by whatever yardstick you choose, streets ahead of the competition. The QL is assessed in our review that begins on page 26, although it must be pointed out that this is a "paper" review based upon a detailed examination of the machine's manual and brief discussions with a few key Sinclair personnel. This approach has been dictated by the fact that at the time of writing (late January) no QLs are available for examination by the press. We have been assured however that a QL will be with us in time to prepare a hands on review in time for our next issue.

The fact that the press could not take away any machines at the time of its launch and that the rumours going the rounds at the moment concern themselves with things that point to difficulties in the initial production of the QL in volume, would tend to suggest that it may be a little while before there are many machines in the field. This makes the first wave of advertising with its claim of 28 days delivery seem optimistic to say the least.

The Spectrum was initially marketed in very much the same way as the QL is being sold at the moment, that is direct response, off the page, mail order advertising. In the Spectrum's case a combination of production problems and component supply problems, meant that a number of customers waited more like 28 weeks rather than days.

Following complaints from a number of people, Sinclair had their wrists slapped by the ASA, the body dedicated to maintaining the legal, truthful, decent and honest aspects of advertising. Slapping wrists however is about all the ASA can do, apart that is from asking a company to alter or, in extreme cases withdraw the offending advert. This action usually happens months after the advert has appeared and after a fair degree of trouble has been caused.

In the case of Sinclair there is no danger of anyone losing any money and anybody not prepared to wait for a Spectrum was given a full refund. Is that the point though? Any company making a 28 day claim should be able to demonstrate that they are in a position to meet the demand that, to borrow a nice legal phrase, could be expected by the reasonable man.

If necessary the company should be required to show such an ability to fulfill demand in a court of law. Failure to do so would result in a financial rather than verbal penalty. Such a scheme would not penalise small, or even large companies, provided that they could demonstrate that any delivery claims were made in good faith.

It is hoped that Sinclair will meet their claims as to deliveries of the QL but I for one would not bet too much money on it. I hope that I am proved wrong – please let me know how long any of you who order a QL have to wait for the thing to arrive.

GARY EVANS

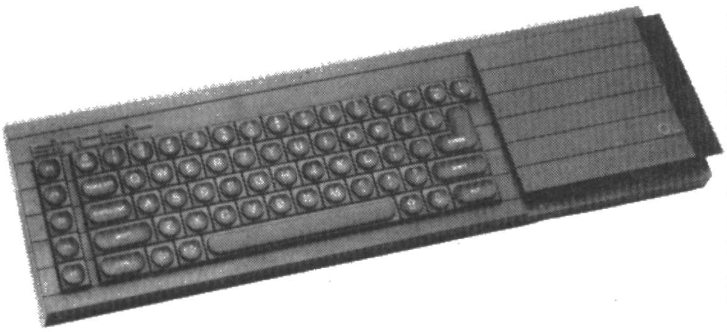
Software pirates clean up

Software houses are starting to shout about piracy. According to the Guild of Software Houses at least £100 million were lost through piracy of computer games during 1983, but the real figure could be 10 times that amount.

According to Nick Alexander, chairman of the Guild and managing director of Virgin Games, 'It is generally regarded that between one and ten pirate copies are made for every game sold'.

The current market for computer games in the UK is £100 million. It is hard to imagine that the pirates are making an equivalent amount (much piracy is of the amateur variety) but this is the figure which the Guild estimates is lost in sales by the 300 British software houses through illegal copying. Some firms feel that if the trend continues they will be threatened with bankruptcy. The legal steps they can take to combat the problem are not yet established, but the industry is said to be laying out its plans.

NEWS NEWS NEWS NEWS



Sinclair has his day

Acorn, Apple, Commodore and the British Broadcasting Corporation all suffered a severe verbal battering when Clive Sinclair used the opportunity of the launch of his new QL to tell a few home truths. The Brian Clough of computing has never been known for his modesty or reticence; neither were in evidence on January 12th.

According to Sir Clive, upgrading the BBC micro to the standard of the QL would cost in the region of £1,500 (Nigel Searle raised a large chart into the air to prove it); Commodore, with their new 264 were 'miles away'; and Apple's Macintosh, with a 68000, the same memory as the QL and a single disk drive offered 'less performance at four times the price'. Vying for their lucrative educational contract, and apparently frustrated at every move, Sir Clive accused the BBC of 'moving in very mysterious ways'.

Needless to say Sinclair has put his money where his mouth is and produced another admirable machine. The Quantum Leap is a new kind of computer likely to open a new kind of market. For £399 the buyer will get one of the best processors available, a first class operating system, two microdrives, versatile networking and interfacing, and four business software packages which two years ago would have cost £400 each. The QL is aimed at 'the technical, professional, and senior student market', but Sinclair claims that they are not leaving the home market: 'the QL is a professional computer for the mass market, designed for more businesslike applications'.

The QL will be marketed for the time being by mail order only. First deliveries are planned for late February. Currently the only machines in existence are pilot stock, and some doubts hang over the company's ability to satisfy what will certainly be a big demand – particularly at a time when there is a worldwide shortage of electronics components. Planned production at the Thorn EMI factories is 20,000 per month by July, but whether QL retains its meaning as Quantum Leap, or gets a new nickname – Quite Late – remains to be seen. Now turn to page 70 for a full review of QL specifications and Superbasic.

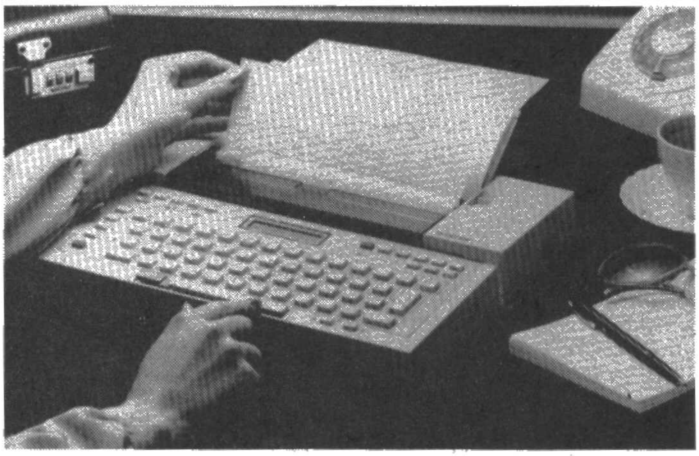
Build your own 6809

Based on the powerful 6809E microprocessor the Microbox is a 'build your own' computer supporting a full 60K of main system RAM, disc controller for two 5.25 inch 40 or 80 track drives, two programmable RS232 serial ports, Centronics printer output port, and parallel keyboard input port.

An additional 128K of on board RAM is partitioned as alphanumeric display, graphic display and silicon disc. The alphanumeric format displays text in 108 columns by 24 lines or alternatively 128 columns by 72 lines. True high resolution graphics of 768 by 576 points are implemented by use of the NEC7220 graphic controller chip. Very high speed drawing is provided by this device through its own hardware vector generation. Primitives supported include points, lines, rectangles, circles, arcs and area fill.

The RAM based silicon disc looks to the system like a conventional floppy disc with a capacity equal to a 40 track single sided single density diskette. However the access time is many times faster.

In addition to the RAM disc an eprom based silicon disc is also implemented on Microbox, this is in the form of a small plug-in EPROM carrier board which will accept four 8, 16, 32 or 64K EPROMS, a maximum of 256K of additional disc capacity. *Microconcepts, 8 Skillcorne Mews, Queens Road, Cheltenham GL50 2NJ.*



The EP44, Brother's portable printer/typewriter/terminal, has lived up to all expectation. All the deficiences of the EP22 have been corrected, and a number of important new facilities are available: not least the ability to send as well as receive, allowing the machine to be used as a dumb terminal.

Print quality is measurably improved, memory storage has been doubled, and on the new machine it is possible to edit all copy stored in the memory. All these facilities, plus an inbuilt calculator (which prints) are available for around £220. Next month *E&CM* will be carrying a full review of the machine, and we intend to test its usefulness as a keyboard for the Spectrum, and as a receiver of data communications via a modem. Readers will have the chance of winning their own EP44 by entering this month's competition on page 72 of *E&CM*.

Prism double launch

With an all star cast and a spectacular light show Prism launched two new products onto the market.

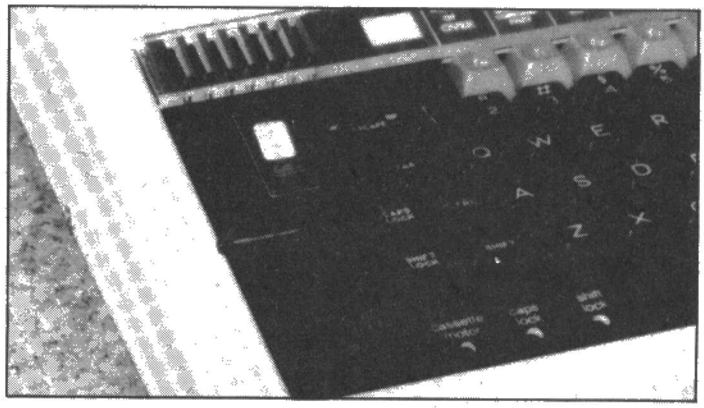
The date was Friday 13th, and not all went as planned. The show began with the launch of Topo and Fred. Topo is a 3' high 'personal robot'. Usually he walks and he talks, but had difficulty doing either on the day. The robots are controlled by infra-red signals, and at the London Hippodrome the transmitters had to contend with an absorbent black wall surface, red lasers, smoke, and an infra-red curtain control. Depending on who you listened to, one or other of these was responsible for Topo's embarrassing immobility.

Also on show for the first time was Prism's first computer, a semi-portable executive machine known as the Wren. With two disk drives, VDU and keyboard incorporated into a single unit, comparisons with the Osborne will inevitably be made. The Wren however has been built first and foremost with a strong communications capability in mind, reflecting the manufacturer's interest in Prestel and Micronet. The Wren offers the ability to send electronic mail to other Prestel customers, can directly download software to Micronet 800, and sends and receives national and international Viewdata information.

Tremendous response to Your Robot

Readers will already have noticed the inclusion of **Your Robot** with *E&CM* for the second month in succession. Originally it was intended to include the free magazine on a quarterly basis, but because of the tremendous response we have had from both readers and advertisers **Your Robot** will be given away free with *E&CM* every month, and will be sold on its own in the near future. The TV advertising campaign which heralded the arrival of **Your Robot** will be continued this month.

HARDWARE



BBC ROM extension socket

Toad Educational Computing has produced a new method of extending the capabilities of the BBC Model B micro with EPROM software. The ROM extension socket sits outside the computer on the keyboard, so changing EPROMs is simple; just lift out one and drop in another.

The extension is a single socket which is fitted easily onto the keyboard casing and plugs into the machine's ROM board. There is no need for any soldering as the unit comes complete and ready to instal. It fits neatly into the area to the left of the keyboard and accommodating it means only cutting out the hole already indicated by the manufacturer's perforations.

The extension socket costs £19.99. *Toad Educational Computing, Dept. PR, 8 Westbourne Grove, Sale, Cheshire M33 1RP.*

Boosting Spectrum sound

Zeal Marketing have designed and are now producing a sound booster for the Sinclair Spectrum. The unit comes complete with leads and a load-save facility which obviates the need for constant plugging and unplugging. A hefty 3" loudspeaker is incorporated and a volume control for more than adequate sound input. The device requires no batteries and all connections to the computer are made externally, thus ensuring no invalidation of the guarantee. The unit is priced at £14.99. *Zeal Marketing Ltd., Vanguard Trading Estate, Storforth Lane, Chesterfield, S40 2TZ.*

Spectrum I/O port – 16 or 48K

Multitron's Spectrum I/O port features 24 I/O lines, single wire sockets on each individual line, and edge card connection. It is possible to construct prototype projects using the computer without the need for any soldered connection. The module is easily programmed using the Basic IN and OUT commands. There is a through bus for peripheral connection, and a comprehensive user manual is supplied. The price is £11.75. *Multitron Electronic Assemblies, 5 Milton Close, Headless Cross, Redditch, Worcs B97 5BQ.*

Dragon prototyping board

A Eurocard size prototyping board for the Dragon and Tandy Colour computers is now available. This double-sided glass fibre board has 1mm holes on a 2.54mm matrix. Available in two versions; standard at £4.95 + p&p + VAT and with a gold-plated edge connector at £6.15 + p&p + VAT. *Steve's Electronic Supplies Ltd., Castle Arcade, Cardiff CF1 2BW.*

Paged ROM on the Beeb

This is an external/internal system comprised of two circuit boards; the first is a small board, mounted internally by plugging into the CPU socket (no soldering required); the second board is boxed and contains 12 x 28-pin sockets and is connected to the first board via a flat cable and connectors. This facilitates inserting and removing PROMS, EPROMS etc (up to 16 Kilo byte in size) without the need to open up the BBC.

If further expansion is required, a second board (also boxed) is available which is connected to the external board and adds another 12 expansion sockets which can replace the first 12 at the throw of a switch. *Micro-Z Ltd., PO Box 83, Exeter EX4 7AF, Devon.*

SOFTWARE

BBC

Doodle Master

A graphics programmer with comprehensive facilities for the definition of up to 8 shapes at a time on the Dragon 4-colour hi-res screens. If no Basic program is resident up to 20 shapes can be defined. Full swopping and movement between the 8 shapes is possible, allowing the 'Old' reloading of a shape into the workspace or 'New' saving of workspace to the selected shape. The package costs £17.95 (cartridge) or £15.95 (disk).

Character Define Envelope Editor

Two new programs from Gem Software released as their Util-1 for the BBC B using OS 1.0 and above.

Character Define allows the user to create 'all manner of user defined characters' and single or multiple character shapes and symbols for use in programs.

Envelope Editor assists in the use of the BBC's sound and envelope commands, with graphic displays to portray the effects created by the modification of the BBC B's pitch and amplitude envelopes.

Both programs are written in Basic, employing 24 and 26K of memory respectively. The price of Util-1 is £9.95. *Gem Software, Unit D, The Maltings, Sawbridgeworth, Herts.*

Design

Beebugsoft's CAD package for the BBC is a screen processor which allows information to be displayed in a format suitable for demonstrations, slide projections, or presentations. Graphs and pie charts can be drawn using the software, and text can be written onto the screen. Twenty user characters may be used to build up large special shapes, they may then be placed anywhere on the screen or redefined as required. Two sets of these characters are supplied with the package: one for electronic circuit design work. Design is priced at £19.00 (disc) or £10.00 (cassette). *Beebugsoft, PO Box 109, High Wycombe, Bucks HP11 2TD.*

DSKEDT

A full diskette sector editor from Premier Microsystems. DSKEDT is a utility program for the BBC user who needs to repair, alter or examine the contents of a disk. The product is supplied as an 8K EPROM, and features a dedicated disk editor, comprehensive commands, and help facilities. Full documentation is included in the price of £19.95. *Premier Microsystems Ltd., 208 Croydon Road, Annerley, London SE20.*

DRAGON 32

Toolkit

This and the two packs listed below are from Premier Microsystems. The Toolkit is a cartridge, utilising only 0.3K RAM; it includes a full screen editor, eight programmable keys, 25 full colour lo-res graphics screens (full colour animation at machine code speed from BASIC is said to be possible), a full range of error handling commands, variable GOTO and GOSUB commands, TRACE, SEARCH, REPLACE, and over 60 new words linked to Dragon Basic. The price is £29.95.

Paintbox

One of a number of new releases from Beebugsoft, Paintbox is a joystick drawing program capable of drawing intricate full colour pictures. It features an 'elastic band cursor' drawing in various widths, and will automatically draw (and fill) circles, triangles etc. The keyboard can be used to insert text anywhere on the screen, using any colour and in any of seven sizes. Paintbox is available at £12.00 (disc) or £10.00 (cassette).

Encoder 09

A full 6809 assembler/disassembler/monitor featuring a full symbolic assembler using standard mnemonics and pseudo opcodes. Basic is used for the editor, so that a full line editor is always immediately available. An auto line generator toolkit is also provided. As a cartridge, Encoder 09 is priced at £39.95, and at £34.95 as a Delta disk. *Premier Microsystems Ltd., 208 Croydon Road, Annerley, London SE20 7YX.*

SPECTRUM

Eastmead Medical Series

An absolute must for all hypochondriac Spectrum owners. Three packages are available: 'The complete guide to medicine', 'First Aid', and the morbidly named 'How long have you got'. The first pack provides information on pregnancy, child development, anatomy, physiology, nutrition, exercise, and a simple diagnostic program. The function of the First Aid Pack is self explanatory, dealing with all the usual type of household disaster, and 'How long have you got' collates and number of variables affecting health such as smoking, drinking, work habits, sex, weight, height, stress – you name it. This one could give you a nasty shock. *Eastmead Computer Services Ltd., Eastmead House, Lyon Way, Frimley, Camberley, Surrey GU16 5EZ.*

ORIC A/D CONVERTER

A. A. Chanerley, B.Sc., M.Sc., describes a sophisticated analogue to digital converter based on the latest, self contained, ADC chip. In addition, he fully describes the subtleties of the machine's I/O facilities.

The Oric computer has a dedicated area of memory for input/output facilities. This area is page 3 on the Oric memory map ie address &0300 to &03FF,255 available memory locations. This is not to say that any other area of memory cannot be used for one's own peripherals, as will be explained shortly. It is just that to prevent clashes occuring with Oric peripherals eg microdrives it is advisable to use the dedicated input/output area.

If one looks at the block diagram of the Oric circuit, **Figure 1,** then it can be seen that the 6502 CPU is connected to ROM/RAM, a 6522 VIA, through which the CPU is also connected to the 8912 sound-generator chip. This IC is also connected to the keyboard interface, and so, the CPU is multiplexed through the 6522, to the beeper, keyboard and printer. The keyboard interface uses the 8-bit I/O port built into the sound chip. This port decodes the columns of the keyboard directly, while the rows are decoded by a 4051, 8-to-1 line multiplexer. The 3 address inputs being connected to the first three bits of port B on the 6522 VIA,PB0,PB1 and PB2. **Figure 5** shows a schematic of the keyboard interface. The 6522 VIA is memory mapped onto the first 16 locations of page 3 ie &0300 to &030F, as shown on the memory map of **Figure 2.** These locations are reflected up page 3 and therefore it is necessary to disable the 6522 when performing a read or write operation to or from an external peripheral. This is possible by

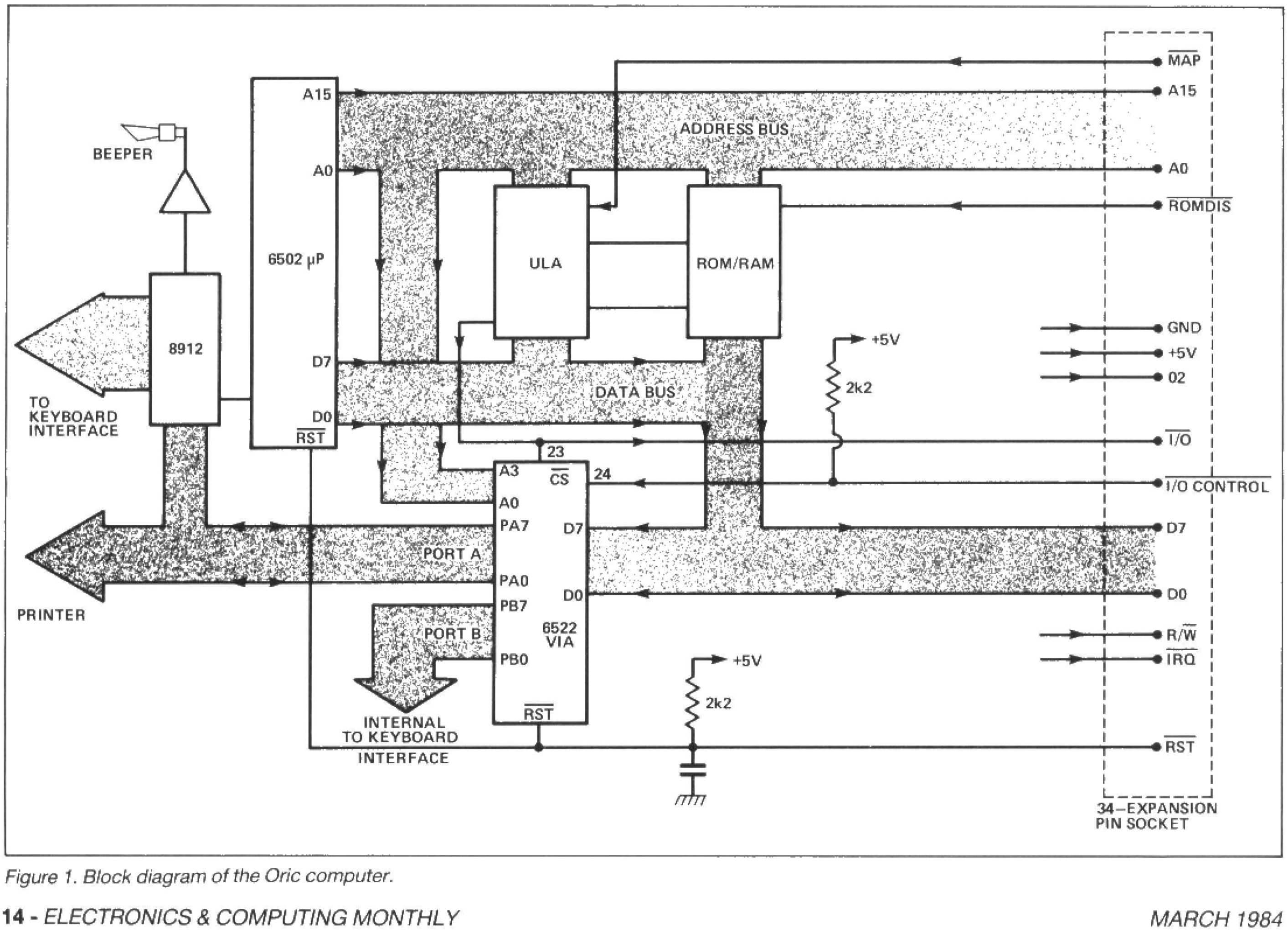


Figure 1. Block diagram of the Oric computer.

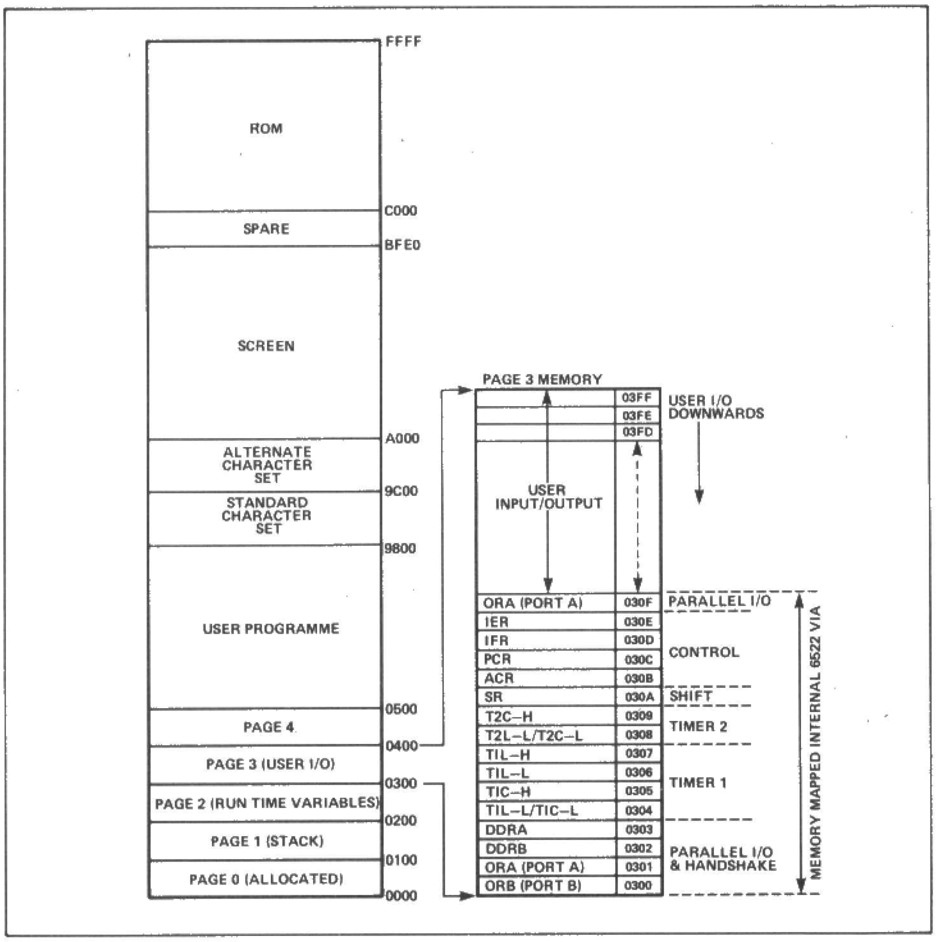


Figure 2. The Oric's memory map.

pulling the I/O CONTROL pin low. This pin, which is brought to the expansion socket at the back of the Oric, is pin 24 on the 6522 VIA and is active high, disabled low. The pin number for the I/O CONTROL input is shown in **Figure 4,** containing the pin-outs of the ORIC expansion socket at the rear of the microcomputer.

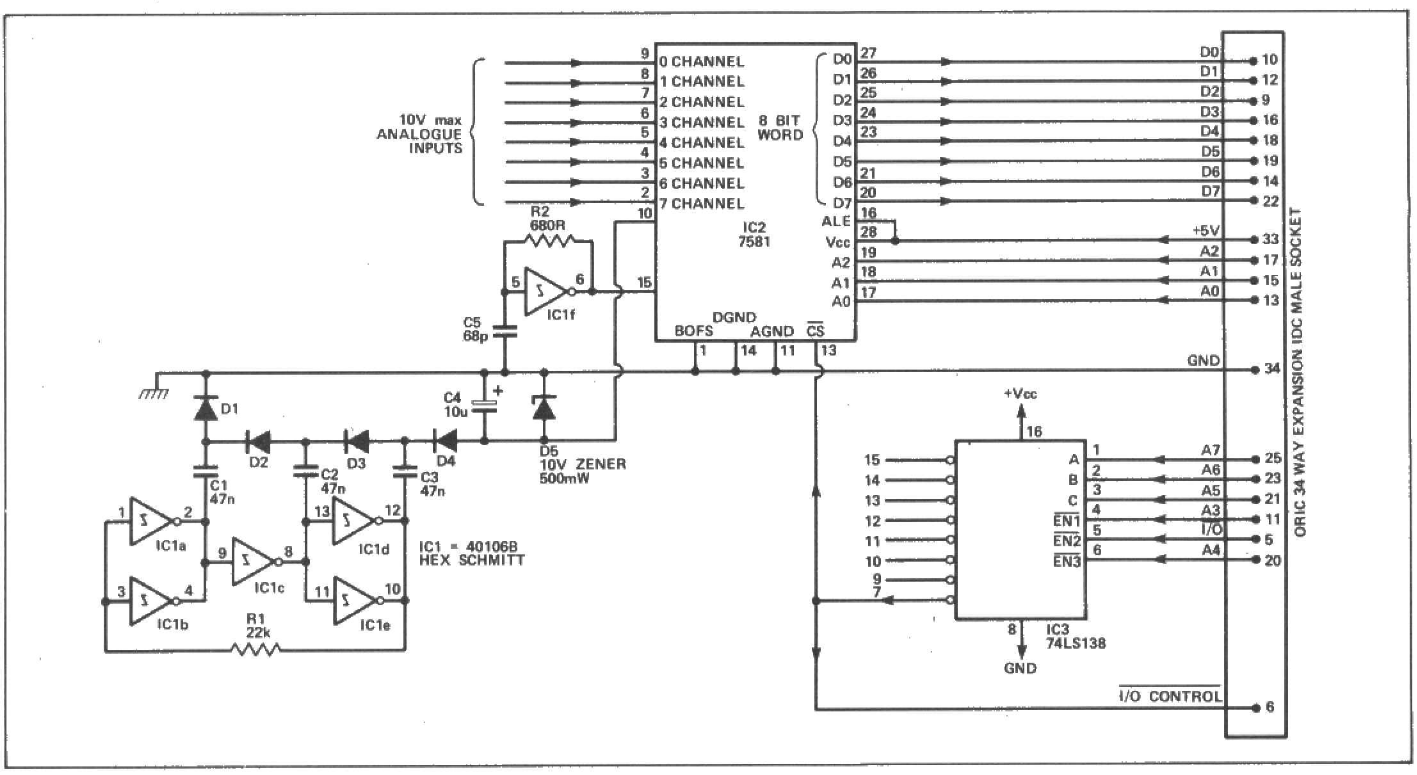
Decoding page 3 of the ORIC map is similar to that employed by Acorn's BBC microcomputer, when using its 1MHz bus. ORIC provide the I/O signal, marked on the expansion socket as pin (5), which is an output, driven low every time page 3 is accessed, this signal effectively decodes the upper 8 address lines, which are A8 to A15. The remaining task is therefore to decode the lower address lines A0 to A7. Users familiar with the BBC microcomputer will recognise that the ORIC I/O output corresponds to the external FRED and JIM outputs which also decode A8 to A15 when pages FCXX and FDXX respectively are accessed. It is an admirable technique for rapid, efficient decoding and is employed in the ORIC version 1.1 and the earlier 1.0 version.

"...BBC owners will see similarities between the FRED and JIM outputs and the Oric I/O".

The I/O signal pin and A0 to A7 are all numbered on the expansion socket as shown in **Figure 4** and via suitable gates can then be used to provide the chip-enable signal (CE) for any external peripheral and to simultaneously disable the internal 6522 VIA by pulling the I/O CONTROL pin low. It should be noted however that the internal 6522 need only be disabled between addresses &0310 to &03FF of page 3, because addresses &0300 to &030F, the 1st 16 addresses are the controlling locations for the operation of the 6522, which can also be used for user input/output as explained below.

The Oric VIA

The 6522 Versatile Interface Adaptor (VIA), is a combination of shifter, timer and parallel input/output (PIO). It is equipped internally with 16 registers which control the chips operation. These 16 registers are mapped onto ORIC memory through loca-



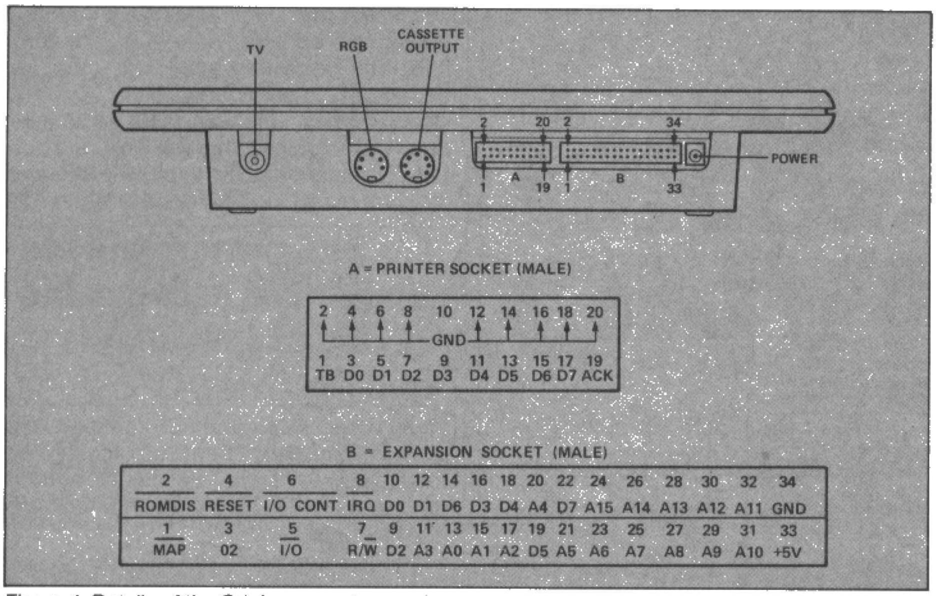


Figure 4. Details of the Oric's connectors.

tions &0300 to &030F and through these the CPU controls the VIA. The exploded memory map shows the locations and control functions of the VIA in **Figure 2.** Fortunately we shall confine our discussion to the 8-bit parallel I/O ports designated port A. Port B and timers are configured internally and therefore can be ignored, port A however forms the data bits of the printer port at the rear of the ORIC and can therefore be configured for output or input.

Directing Data

To direct data to the printer socket data-pins only requires a POKE statement. The memory map shows that locations &0303, &0301 and &030F are used for input or output for port A. Location &0303 is mapped onto the DDRA, the data-direction -register for port A. This register, under software control tells the VIA whether port A should be an output or input. By POKEing the word 255 into DDRA, all 8 bits of port A will be set for output. By POKEing 0 into DDRA, all 8 bits will be set for input. Any of the 8 bits can be individually set for output or input by POKEing in the necessary control bit or either 0 or 1. For example, to POKE data of decimal 20 into port A, will be as follows:

```
10  POKE #0303,255:
    REM make port A output
20  POKE#0301,20:REM output data
```

and bits 2 and 4 ie pins (7) and (11) of the printer port will be set high, all the rest will be low. The 20-way printer pin-out is shown in **Figure 4.** In order to program for input:

```
10  POKE#0303,0:
    REM make port A input
20  Y=PEEK(#0301):
    REM read input data
30  POKE#0303,255
          :REM reset via as output
          :REM otherwise keyboard
          :REM will be disabled
```

Bearing in mind that I/O goes low when page 3 is accessed and the R/W line is available, latching can be performed at output.

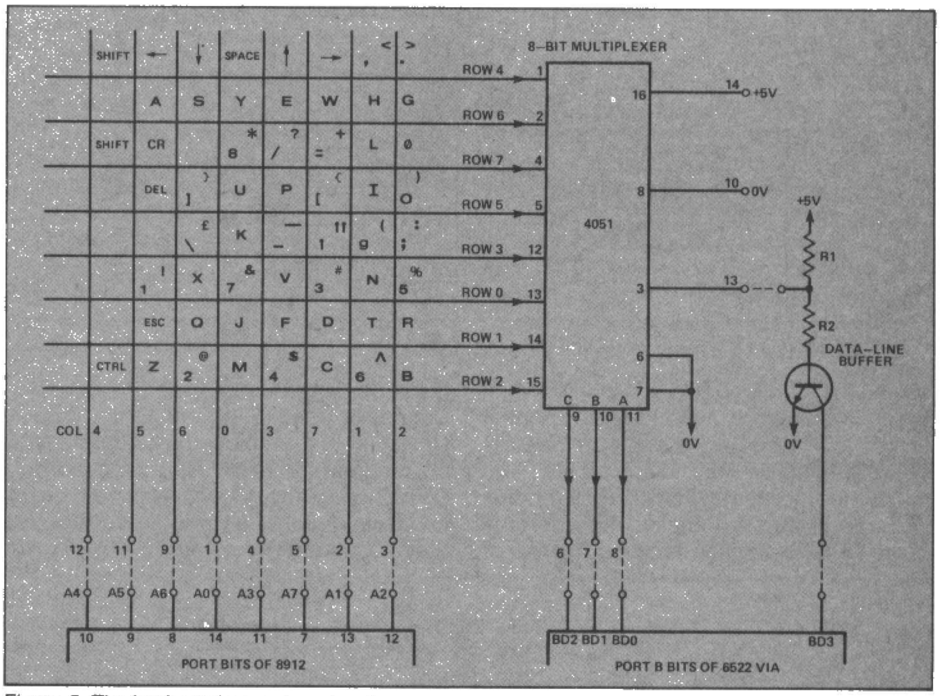


Figure 5. The keyboard scanning circuit.

Other memory locations

Memory locations BFE0 to BFFF have in the past been used for user defined peripherals, by using the MAP and ROMDIS pins. The purpose of the MAP pin, which is an active low input, is to modify the operation of the internal RAM or ROM. If MAP is pulled low when locations in ROM are being accessed, then the internal ROM is disabled and the internal RAM enabled giving up to the full 64K RAM with the 48K machine. This is used by the ORIC microdrives, hence clashes may occur with user defined peripherals in these areas. If RAM is being accessed when MAP is pulled low then the internal RAM is disabled and external memory or other add-ons may occupy the data bus. So additional paged ROMs, for example may be designed onto the ORICs expansion socket. Timing is important, MAP should be active low for about 250nsecs and should commence 80-100nS preceding the rising edge of 02. More details can be obtained from a new book by Ian Adamson, The Companion to the ORIC 1. Where the author of this article wrote the I/O chapter.

An Oric ADC

The ADC chip that is at the centre of this design is relatively recent, but has already been extensively used in interfacing other computers. It is a virtually self contained 8 analogue input, 8 bit, A-to-D converter chip, which requires a −10 Volt reference typically, but can be + or − 25 Volts, a 5V TTL supply and a 1MHz to 1.6MHz clock to synchronize conversions. See circuit diagram **Figure 3.**

Internally the chip has a multiplexer to select the analogue channel, an on board D-to-A converter, a successive approximation register, an 8x8 RAM for storing currently converted data, tri-state buffers which when enabled will place the data onto the data bus of the computer. The block diagram is shown next month. Each of the analogue channels takes 80uS to convert, that is on a 1MHz clock, therefore a complete 8-channel conversion takes 640uS. Each time a channel conversion is complete the data-byte is loaded into the appropriate internal 8x8 RAM location, ready for output onto the data bus when the buffers are enabled by a low CS (chip select) signal. There is also a status pin (12), which goes low every time a channel conversion is complete and this provides real-time applications. A0,A1,A2 are used to address the 8x8 RAM locations and are controlled by the operator from software, in this case by PEEK statements to read in the data. There is an address latch enable (ALE) which is permanently tied high.

Next month: Construction details.

Battle of the sec

Mike James unravels some of the complex ideas and techniques in

The BBC Micro is a remarkably well designed machine that has always been seen as the centrepiece of a computer system rather than a complete system in itself. This is not to say that the BBC Micro cannot be used on its own for a good range of applications but criticisms of lack of memory are often heard. In practice the BBC Micro is fast and the shortage of memory really only becomes acute when using Mode 0 or Mode 1. However 'fast' is a relative term and no matter how fast a machine is there will always be someone wanting it to go faster! In the past the only solution to the 'go faster' problem was to scrap the whole machine and start again! This was the fate of ENIAC, EDSAC etc but not the BBC Micro. Rather than design a machine that would become redundant as soon as improved versions of the traditional eight-bit micros and the even faster 16/32-bit devices became available, Acorn realised that the most important part of any system was an I/O processor and this is what they designed.

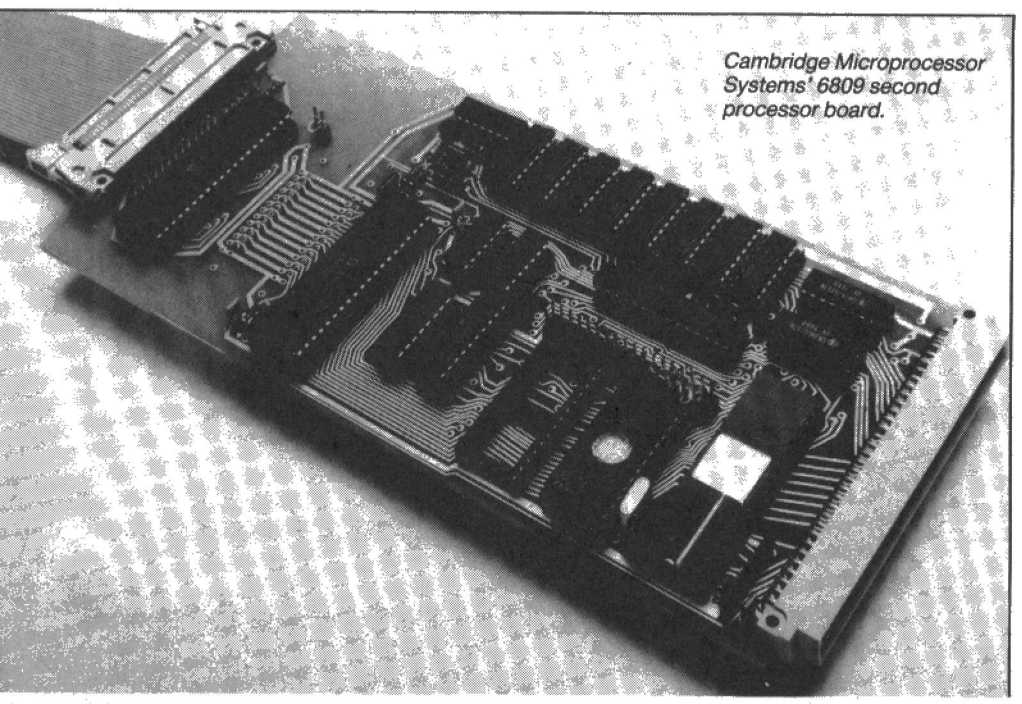
How simple is a computer

Seeing the BBC Micro as nothing more than an I/O processor may be a point of view that is difficult to accept – almost demeaning of the BBC Micro's normal hallowed position as one of the best machines around. However this is a misunderstanding of just how complicated an advanced I/O processor is and just how simple a computer is. Early computers communicated with the outside world via serial and parallel interfaces. Driving these interfaces was comparatively simple but then the responsibility for actually displaying the output was left to complex devices such as VDUs and printers. With improving technology more and more of the originally separate I/O devices have become built into the computer. Today it is quite natural to expect a personal computer to include the necessary hardware to produce not only a text display but a full colour graphics display, a disc drive interface and so on . . . But the part of a computer that is responsible for its raw computing power is still the same. The quality of the CPU governs the speed of processing and the amount of memory governs the size of program that can be run (without special techniques).

If you ignore the problem of I/O it has never been easier to build a computer. All you have to do is assemble a few (and the number decreases every day) RAM chips for the main memory around a microprocessor of your choice. Add a small amount of ROM or EPROM to get things started and apart from the few small chips that are necessary to make it all work together (the 'glue' in current jargon) that is all you need. Of course with no way of getting data in or out of the resulting computer it won't be at all useful.

Once you have realised that the 'computational' part of a microcomputer is a very small simple component of the whole system you start to question the need to throw the whole machine away when all that is required is a change of CPU and perhaps some additional RAM. This is a question that was first answered by the APPLE II with its range of add-on co-processors. There are a great many CPU cards for the Apple II and if it wasn't for the fact that the Apple's I/O functions, graphics, disc etc, are rather out of date the Apple II would be as viable today as it was when the 6502 was an exciting new micro. The point is that even I/O processors eventually become old technology. The BBC Micro is one of the best I/O processors available. It is so good that a great many BBC Micros have been bought to be used as colour VDUs. Its action as an I/O processor is far from limited to just an excellent graphics and text display. It can handle discs, a sound generator, an A to D converter, a speech synthesiser, a parallel port, a serial port, a cassette system and a local area network for any other computer that is connected to it. If you include electronics housed in other boxes then the list can be increased to include a Teletext and Prestel Interface. With such a range of possible I/O devices and operations it is amazing that the BBC Micro can be bothered to run a BASIC program at all!

". . . fast is a relative term . . . there will always be someone wanting their machine to go faster . . ."



Cambridge Microprocessor Systems' 6809 second processor board.

Tubes of communication

Now that the idea of adding a second processor to the BBC Micro seems like an excellent one, the problem of exactly how to do it remains. Obviously we could go back to the original idea of using a serial interface. This would turn the BBC Micro into the equivalent of the VDU and as long as the data that was passed to and from the second processor was limited to small amounts of text and graphics commands this works well. If you increase the data flow to include data files then the weakness of the serial interface becomes obvious. The speed gained from using a second processor would be completely wasted if communication between it and the I/O processor were slow. The trouble is that the form of the interface between the two components of the system has to be very fast but it also has to be very simple to keep the component, count and hence cost, of the second processor low.

The most obvious method of passing data at speed between two computers is a parallel interface but there are a variety of design features that can be incorporated into a simple parallel interface that can be

ond processors

volved in the design and use of second processor for the BBC micro.

critical to the overall performance. The BBC Micro has three possible connection points for inter-processor communication – the user port, the 1MHz bus and the Tube. Of the three only the Tube is intended for second processor connection but there is not reason in principle why the other two should not include other processors among the list of equipment that they can be used to interface. Only the Tube however has any hope of realising a high data transfer rate because it is a direct connection to the full speed (2MHz) system bus.

Back-to-back PIA's and Acorn's Tube ULA

From the BBC Micro's point of view the Tube consists of 32 memory locations between FEEE0 and FF00. It is not difficult to see that with the right hardware this area could be used to transfer data to and from another micro using very simple software. The most obvious hardware arrangement is to use a pair of PIA's or VIA's connected back to back **(see Figure 1).** This provides two uni-directional parallel ports, one from the BBC Micro to the second processor and one from the second processor to the BBC Micro. One advantage of using PIAs is that they can be configured to perform automatically the handshaking necessary to transfer data asynchronously. But even with automatic handshaking a PIA cannot achieve the transfer rate of 2Mbytes per second that the clock rate might lead you to believe. Problems with testing and resetting flags or servicing interrupts slow the transfer rate to much less than 1Mbyte a second.

Although the back-to-back PIA solution is suitable for interfacing non-6502 second processors it does have some shortcomings. As will become apparent after the description of the way software is shared between a pair of 6502 processors there are good reasons for needing a higher performance. Of course the solution is a custom-designed ULA to implement a more sophisticated parallel interface for the Tube. This is exactly what Acorn have done and the resulting ULA gives the Tube an incredible (or should it be credible?) performance. The Tube ULA provides four bidirectional byte parallel ports and in itself provides roughly four times the data transfer capacity of back-to-back PIAs. Each port has its own flags and interrupts to control data transfer automatically. To smooth out any temporary hold ups in data transfer there is a 24-byte FIFO (First In First Out) queue for use by the busiest channel (associated with OSWRCH, see later) and a number of smaller FIFO queues for the other less pressed channels. These facilities make it possible for concurrent processes to communicate without interfering with each other and this is obviously an important consideration when the second processor is going to be a 32-bit super micro!

The quality of the CPU governs the speed of processing . . . size of memory determines the size of program that can be run.

As well as considerations of performance a ULA can solve a number of problems of reliability. Two machines that are trying to exchange data asynchronously can be prone to unforseen 'race hazard' conditions. A race hazard is a situation that arises when the state of a logic circuit depends critically on the time that the pulses arrive. A circuit that normally functions perfectly can occasionally 'lock up' or even oscillate due to the occasional chance co-incidence of pulses. In the design of custom interfaces this sort of problem has to be completely removed rather than be made unlikely because the vast number of data transfers will cause the machine to crash every few hours even if the race hazard problem is made incredibly unlikely. A second area that a custom interface can deal with is bus isolation. If for some reason the second processor is powered down it is important that the I/O prcessor is still able to function. This can only be achieved by providing sufficient isolation between the two machines. You should by this point be able to see how much advanced technology has gone into producing Acorn's Tube ULA an investment that must pay off as their range of second processor products expands.

Software through the Tube

The hardware characteristics of a fast inter-processor communication channel are almost irrelevant to the programmer provided that is that it is fast, reliable and supported by low level byte or block transfer software. From the software point of view there are two distinct situations:

> the I/O processor and the second processor are the same micro and there is a measure of interchangeability between them

and

> the I/O processor and the second processor are different and clearly defined as such

There is no doubting that the first situation is the more complicated and this will be returned to later.

If the intercommunicating processors are different there is generally no confusion of function between them – the I/O pro-

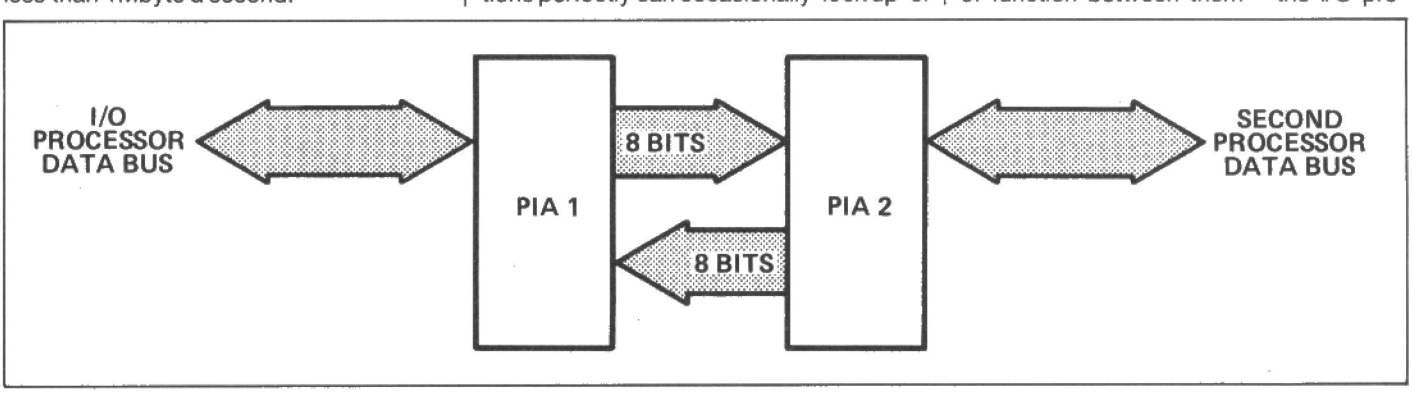


Figure 1. The most obvious hardware arrangement involves back to back PIAs.

cessor serves the second processor as and when it requests I/O. For example, if the second processor is a 6809 then the operating system that it runs, FLEX say, will be stored in the second processor's own memory. As I/O requests are generated by applications software running under it they are passed across the Tube to the I/O processor. In the simplest case the I/O request is nothing more than a single character that is destined for the TV screen. At its most complex the operating system may be reading or writing a fule and a disc transfer is involved. However it is important to realise that in this case the file organisation is the responsibility of the operating system in the second processor. All that it requests from the I/O processor is a particular sector. From the software point of view, once the operating system has been modified to use the I/O processor the second processor behaves like any other machine of its type.

The picture with a 6502 second processor is not so simple. As both processors are the same they use the same operating system – the BBC MOS. It would make little sense to transfer a complete copy of the MOS into a second processor, after all one of the advantages of the second processor is its extra memory and using it up storing a second copy of 16K of software would be very wasteful. Fortunately the BBC's operating system is not only very well designed (see "The BBC MOS" in December 1983's issue of *E&CM*) but is well designed with the Tube in mind. The fundamental I/O calls, for example, OSWRCH (Operating System WRite a CHaracter), are all vectored through RAM locations and so it is possible to intercept all such calls and replace the subroutines that would normally implement them by Tube software. That is, when OSWRCH is called the character is not passed to VDU drivers in the second processor but to VDU drivers in the I/O processor via the Tube. In this way the second processor can be 'fooled' into thinking that the entire MOS is resident when in fact there is only a few K of Tube software. Acorn's 6502 second processor is a triple speed (3MHz) device and the tube software uses only 2K of the 64K of RAM but it provides that same services that the MOS would. Notice that as the screen memory etc are all part of the I/O processor's memory map the full 62K of RAM is available for user programs!

Another desirable feature of a second 6502 processor is that a program that works in the standard BBC Micro should transfer to the second processor without modification. If you have followed Acorn's recommendations about using MOS calls to manipulate I/O processor memory areas rather than direct access your program will run in the second processor without knowing where it is! For example, if your program uses any of the I/O device registers by way of instructions like:

```
LDA &FE60
```

which loads the A register from the B side of the user VIA then your program will only work if it is loaded into the I/O processor. The reason for its failure in the second processor is simply that there is no VIA at address &FE60 in the second processor's memory map! However if the VIA had been accessed using:

```
LDA #&96   OSBYTE code to read I/O
           data
LDX #&60   Offset from start of I/O area
JSR &FFF4  jump to OSBYTE
```

and the data is returned in the X register then the program would run in either processor because it would call the appropriate version of OSBYTE depending on which side of the Tube it was on. If you always use OSBYTE calls to manipulate fixed I/O processor addresses then your programs can take advantage of the increased speed and larger memory of the second processor without altering a single instruction.

"Seeing the BBC micro as nothing more than a I/O processor may be a point of view that is difficult to accept".

Some real second processors

To bring this brief introductory look at second processors for the BBC Micro to a close it is worth giving a list of currently available and soon to be available products. Each of these products will be reviewed in detail over the next few months.

6809 – the 6809 is my favourite processor (apart from the 68000, that is!) and the second processor card from Cambridge Microprocessor Systems (11 St. Margarets Road, Girton, Cambridge CB3 0LT. Tel: (0223) 276791) provides not only a 6809 with 64K of RAM but also my favourite operating system – FLEX. The Tube interface is by a pair of back-to-back VIA's and custom written software. Once booted up the BBC Micro's discs will read and write standard FLEX format discs (a special utility program will read and write BBC-format discs under FLEX). The card is small enough to fit inside the BBC Micro's case. It also has a standard Acorn bus interface (if you remember, it was with this bus that Acorn started back in the days before the Atom!) and can be rack mounted with additional interface cards. It is ideal as a development system since a stand alone single board 6809 is also available to run production software. Price – £249 for the second processor plus £10 for an interface cable plus £130 for FLEX.

6502 – Acorn's own 3MHz 64K RAM 6502 is to go on sale in April and (of course) uses a Tube ULA interface. It is fast, reliable and compatible with existing software. Price – £195.50.

Z80 – There is a choice of three Z80 second processors: Acorn's own will become available in May. Once again it will use the Tube ULA. Price – between £300 and £400.

Torch are the manufacturers of perhaps the best known Z80 second processor in the form of the Torch Z80 disc pack. This consists of a 64K Z80A processor in the same box as a pair of disc drives. The disc drives are connected to the BBC Micro and can be used directly by the BBC Micro's DFS (Disc Filing System) or via CPN, Torch's own version of CP/M. Price – £730.

The final Z80 second processor is from Watford Electronics who are building up quite a reputation for enhancing the BBC Micro. This is slightly different from the other second processors in that it includes its own disc controller chip and therefore doesn't make use of the BBC Micro as a disc controller. However the disc controller is double density and will provide twice the BBC Micro's normal storage capacity if double density drives are used. It is also different in that it uses the 1MHz bus rather than the Tube for inter-processor communication but as there is no need to transfer disc data between the two machines the 1MHz bus is more than fast enough for the amounts of data involved. The Watford unit also uses CP/M proper and can handle 3½", 5¼" and 8" floppy discs. It is packaged as a separate unit in a half height floppy disc drive case. Price – £399.

"Watford's 2nd processor includes a disc controller chip".

"The hardware characteristics of fast interprocessor communication are almost irrelevant to the programmer".

Thanks are due to Acorn, Cambridge Microprocessor Systems, Torch and Watford Electronics for much help and information.

Next month: Mike James takes a more detailed look at some of the second processors mentioned here.

Suitable for MTX 500

The ultimate speech synthesiser

The speech synthesiser featured in this article can be connected to any computer which has the capability to drive a printer with a parallel centronics interface. This facility is found on many home computers including the following – BBC Models A and B, Acorn Atom, Oric-1, RML 380Z, Tandy machines, Dragon 32, Memotech MTX500 and MTX512 and the Sharp MZ700. However, this article concentrates on only two of these machines – the Dragon 32 and the BBC.

A sophisticated speech synthesiser that can be used with any computer featuring a centronics interface. The circuit also provides four levels of inflection. Design by Russell Vowles*.

*L B Electronics

The speech chip

The synthesiser described is based around the General Instruments SPO256-AL2 Allophone Speech Synthesis integrated circuit which incorporates the following functions:-

1. A programmable digital filter which simulates the human voice tract giving "natural" sounding speech.
2. A 16K ROM which contains the data including volume, pitch and duration for the 64 allophones (sounds) which can be produced by the chip.
3. A micro controller which controls the flow of speech data to the filter and the linking of allophones together to form words and phrases.
4. A pulse width modulator which creates a digital output which can then be filtered to give an analogue output used to drive an audio amplifier.

Centronics interface

The centronics interface provided with the host computer provides 8 data lines, a strobe output (which tells the device connected when there is data ready on the data lines) and an acknowledge input which is taken to 0V when the device is ready to receive new data.

Circuit description

The circuit consists of 6 basic stages as shown in **Figure 1.** The 8 bit data bus, strobe line and the acknowledge line are provided by the host computer and the connection of these to the synthesiser from the Dragon 32 and the BBC is shown in the circuit diagram **(Figure 2)** and in the connection diagrams **(Figures 3 and 4).**

Valid data is only present at the centronics interface for a short amount of time and so this data has to be "captured" by the circuit so that the data is available when the strobe line is pulsed to 0V and at this instant the 8 bit data latch IC1 "captures" the data present on these lines which then becomes available on the outputs of IC1 (Q0-Q7). Six of these outputs are used as a 6 bit address for the speech chip IC3 and select which allophone is to be spoken.

The remaining outputs Q6 and Q7 are used to control the frequency of the clock generator IC2 and hence the tone of speech, allowing inflection to be put into

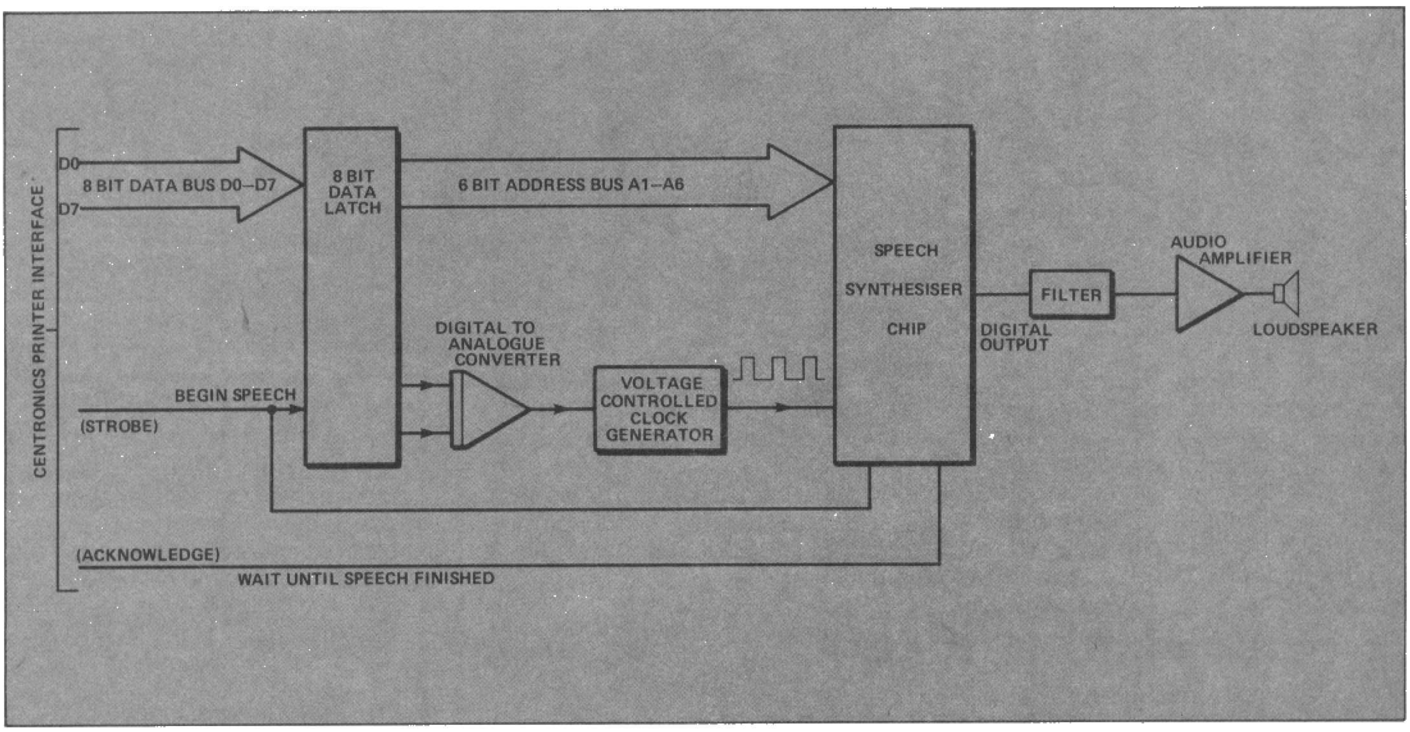


Figure 1. The system's block diagram.

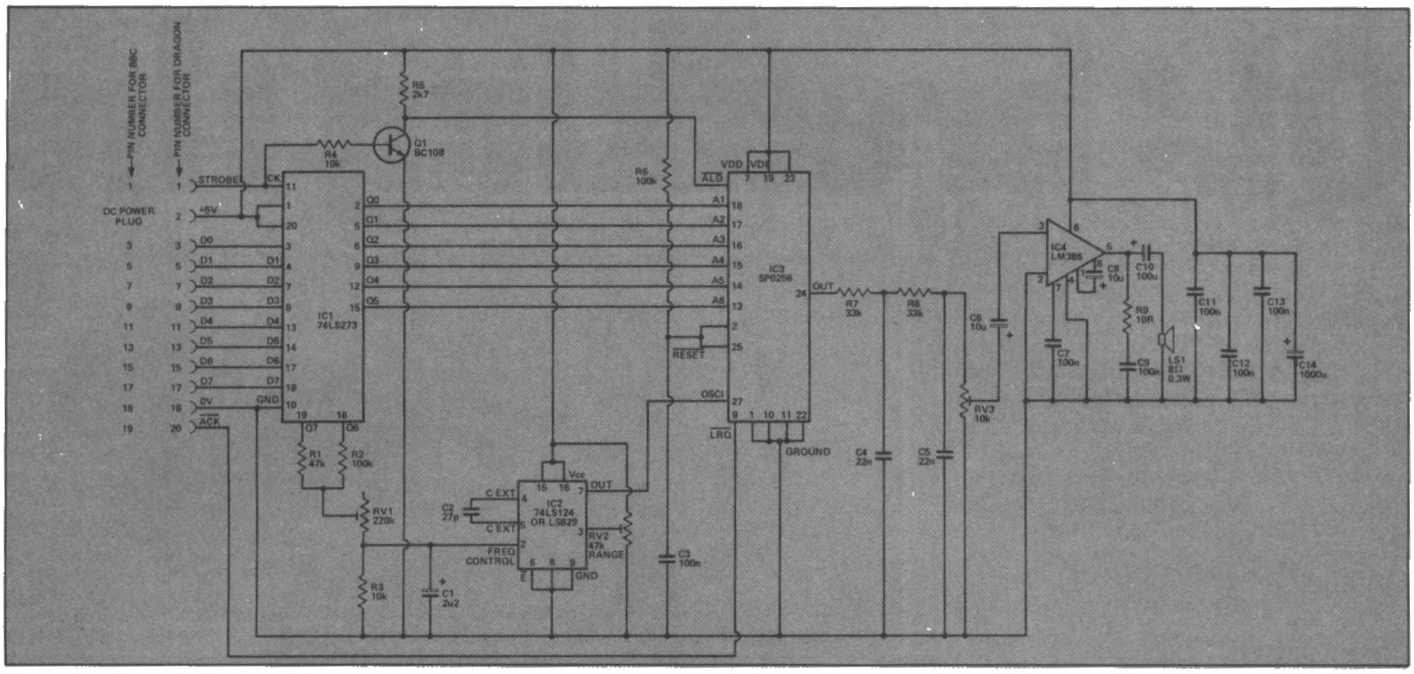


Figure 2. Full circuit diagram of the speech synthesiser.

words. This is achieved by using a 2 bit digital to analogue converter consisting of resistors R1, R2, RV1 and R3. Potentiometer RV1 is used to set the depth of inflection and capacitor C1 is used to prevent rapid changes in clock frequency which could result in poorer speech quality.

The clock generator is based around IC2 which is a voltage controlled oscillator whose frequency is determined by capacitor C2, the voltage on its range input (Pin 3) and the voltage on its frequency control input (Pin 2). The voltage on Pin 3 is used to vary the basic tone of speech, and this is derived using RV2 as a potential divider. The voltage on Pin 2 comes from the digital to analogue converter and controls inflection as described above. The output of IC2 is taken from Pin 7 and is a square wave of a frequency around 3.1MHz which is fed to Pin 27 of IC3.

The speech chip IC3 is a very complicated device. However, it is relatively simple to use, only requiring a few control lines to make it "speak".

After data has been latched into IC1 the strobe line returns to 5V. This signal is inverted by Q1, R4 and R5 causing Pin 20 of IC3 to go to 0V. Pin 20 is the ADDRESS LOAD input of the speech chip and when taken to 0V it causes IC3 to load the address held by the 8 bit latch into the chip's internal address register. This also triggers the production of a speech output and the LOAD REQUEST output (Pin 9 of IC3) changes to +5V until the speech output ceases. While LRQ is at +5V the computer's acknowledge input is held at +5V causing it to wait until speech has stopped before more data is sent to the synthesiser.

Resistor R6 and capacitor C3 cause the RESET inputs of IC3 to go to 0V at switch on, clearing the speech chip's internal registers.

The digital speech output is available at Pin 24 of IC3. This digital output is filtered by the low pass filter consisting of R7, C4, R8 and C5, producing a signal at the volume control RV3 which is ready for amplification.

The audio amplifier is based around IC4 which is an LM386. This device can operate down to a voltage of 4V and can directly drive an 8 ohm speaker. Capacitor C6 is used to prevent any direct current reaching the amplifier. Capacitors C7 and C9 and Resistor R9 add stability to the circuit while Capacitor C8 sets the amplifier's gain at around 200 times.

Capacitors C11-C14 decouple and smooth the power supply, stopping any interference with the host computer.

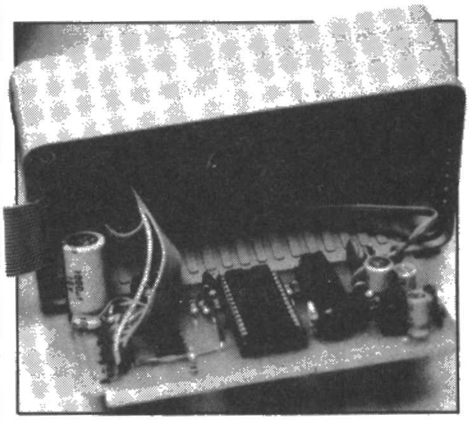
Allophones

The speech synthesis integrated circuit used can reproduce 64 different sounds or allophones (and in this circuit four levels of inflection are available). The circuit allows the user to synthesise any English word by contatenating individual speech sounds.

Other methods of speech synthesis allow the user to use only pre-defined words. These methods are really only refinements of digital recording and require vast amounts of memory. These methods are costly and not very versatile, but do have the advantage of marginally better speech quality compared to the Allophone system which is inexpensive and has an unlimited vocabulary.

"... by adopting a Centronics interface the design is compatible with many machines ..."

A list of allophones and examples of their use was shown in last month's *E&CM* and it can be noticed that there is not a direct match between letters of the alphabet and the allophone sounds. This means that to produce words they must first be broken down into sounds. For example the word "HELLO" consists of the sounds – HH, EH, LL, OH, which are represented as HH1, EH, LL and AW using the allophone set.



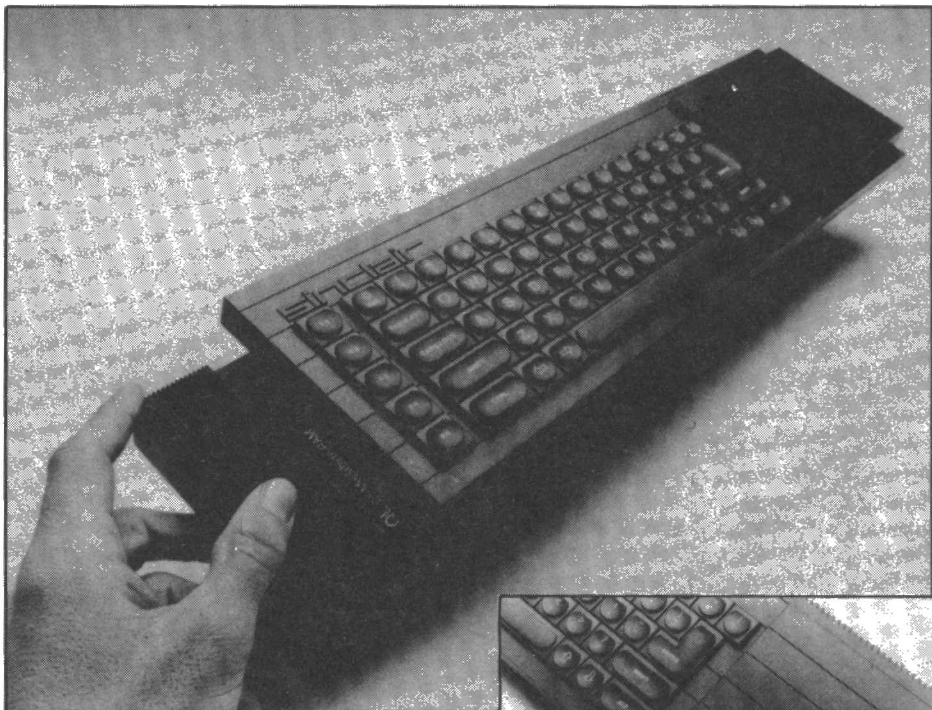
Internal view of the synthesiser.

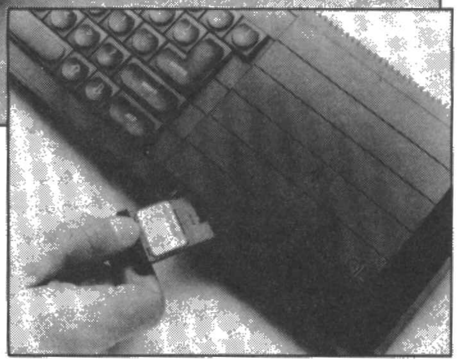
Some experimentation is required to make the word sound as realistic as possible as a speech sound changes depending on whether it occurs at the beginning of a word, next to a vowel, or at the end of a word. Therefore, the allophone set provides different versions of some sounds, eg HH1 and HH2, KK1 and KK2. Varying the pitch (adding inflection) of the sounds in a word also makes the words sound less monotonous and more realistic and the circuit caters for four levels of inflection.

Next month we present full construction details and information about the complete kit of parts to be sold by LB Electronics.

QL: breaking the

By installing two Microdrives into an 8/32-bit computer with a full-travel keyboard, multi-tasking operating system, and including four very sophisticated software packages, Sinclair has swept aside all competitors and slashed the price of a functional business computer from £1200 to £400. The BBC micro has now been declared officially obsolete and the IBM PC Junior an expensive executive toy. So much for the superlatives: what of the facts? Let's take a look at the guts of the machine.





Above, the Sinclair QL with a prototype of the forthcoming 0.5Mbyte RAM pack. Left, inserting a Microdrive cartridge into the QL.

Atoms, Electrons and Spectrums: all have been upstaged by a Quantum Leap. The jargon of physics has finally leant itself to the accurate descriptions of a computer: the Sinclair QL is one unit of magnitude beyond its predecessors.

Hardware

This is a paper review, in that no machines have yet been distributed to the press; however, we do have complete specifications of both hardware and software, as well as a very hefty manual (marked PROVISIONAL throughout).

The QL is packaged in the manner of previous computers from the Sinclair stable: small and black (but not too small). Dimensions are 138 x 46 x 472mm. The keyboard is of professional quality, with 65 sculptured, bucket shaped keys, including 5 function keys, 4 cursor control keys, and a full-length spacebar. To the left of the keyboard are the two Microdrives, each with a minimum capacity of 100 Kbytes and average access time of 3.5 seconds (ie slightly better specs. than the Spectrum version, with which these are incompatible). Spectrum cartridges can be reformatted to work on the QL drives.

The development of the Microdrive must have been the key to manufacturing such a cheap business system. With neither the speed or the memory capacity of discs, they are at this level an efficient and economic option. Using the Microdrive expansion slot on the QL up to six drives can be stacked externally – giving 800K maximum storage. Any program running over 100K can be switched from one drive to the next.

The QL has 128K of RAM, expandable externally to 640K. 32K of RAM is dedicated to the screen bit map when using hi-res graphics. The ROM, in which is held Sinclair SuperBASIC and the QDOS operating system, has a capacity of 32K, expandable via cartridges to 64K.

Travelling further into the depths of the machine, we find two processors. The CPU is a Motorola 68008, with an 8-bit data bus and 32-bit internal architecture (see box for further details) and capable of addressing 1Mbyte of memory. A second processor, an Intel 8049, controls the keyboard, generates sound, and acts as the RS232C receiver. Two Sinclair designed semi-custom chips control memory, and Microdrives, real-time clock, LAN, and RS232C transmission.

Sound and vision

Information on sound is strictly limited: we can only hope that lightening doesn't strike twice (reference the mute Spectrum). All we do know is that 'sound is generated according to a series of preset sound types specified by parameters to the system'. Examination of the BEEP command however reveals considerable versatility, with control of duration, pitch, fuzz, rate and 'wrap'(?).

The character display is 85 x 25, with a choice of character sets available; the number of characters per line can be reduced for TVs with poor resolution. Two graphics modes are available: Hi-res, in 4-colour with 512 x 256 resolution; and Lo-res, with 8-colour 256 x 256 resolution. More colours can be obtained by used a stipling effect. 32K of RAM is occupied by the Hi-res graphics.

Expansion

Everything is there except a Centronics interface – an omission Sinclair may come to regret. Oh, and there isn't a cassette

price barrier

socket – but who needs it with two Microdrives!

Starting from left to right, we first see two phono sockets marked NET. A Local Area Network (needless to say this one is called QLAN) is now seen as a necessity in any serious computer, and especially one that wants to win the BBC contract. QLAN is capable of linking up to 64 QLs or Spectrums in a 100Kbaud network. Next is the power socket, then, a DIN type RGB socket which connects to mono or colour monitors (four grey scales are available in mono). There are two RS232 sockets; one presumes that the first is intended for use with a printer, the other with a modem.

Inside the 68008

Sinclair will tell you – with not a little poetic licence – that the main processor in the QL is a 32-bit device. The Motorola 68008 does have a 32-bit internal address structure, but the data bus is only 8-bits wide, slowing down operation considerably. The true 32-bit chip exists only on the drawing board (Zilog's) and samples will not be available until mid '84.

Readers may be more familiar with the Motorola 68000; this is a 16/32 bit mpu (16-bit I/O, 32-bit internal architecture) and is known conventionally as a 16-bit device. This chip has a cleverly divided stack with four sections, each with the capacity to address 16Mbytes, raising the total addressable memory map to 64Mbytes – an ideal programmers tool!

The 68008, while equipped with only 8-bit I/O, is nevertheless one of the fastest and most sophisticated processors available. It too is an excellent programming device addressing a 1Mbyte memory map: quite sufficient for any application on the QL. 70,000 elements are incorporated onto the chip. The 32-bit architecture permits very accurate and quick arithmetic operations (minimum instruction time is 0.5μsec). This power and speed will present ideal opportunities for machine code programmers and we can expect to see the quality of both games and applications software to improve dramatically. The availability of such a high standard of processing power will doubtless persuade many serious programmers to chuck out those Z80 and 6502 based machines; they will have to learn a few new tricks, but we will see what we can do at *E&CM* to aid the process of change!

Both are very versatile, with a choice of baud rates between the ranges of 75 and 19,200, or transmit/receive in full duplex at 7 rates up to 9,600 baud. Sinclair say that a parallel interface (Centronics no doubt) **will** be available in the future.

The QL may be intended first and foremost as a business machine, but the joystick ports have not been left out. There are two BT600 series sockets for joysticks to be used in either games, cursor control, or graphics. The final slot at the back of the machine accepts ROM cartridges; doubtless the design is identical to that of Interface 2 on the Spectrum, and the cartridges themselves appear to be of the same format.

To the right of the machine (from the back) is the main expansion slot, the major purpose of which will be to connect up the handsome 0.5Mbyte RAM board, which has already been developed.

What of future expansion? The 68008's 1Mbyte linear address capability means that the potential will be almost unlimited. Sinclair are already developing, or intend to develop, a number of enhancements, including the 0.5Mbyte memory board, a Pascal compiler, 68000 assembler, terminal emulator, A/D interface, Winchester interface, modem, parallel printer interface with multi-channel sound generator, and IEEE-488 interface. Bets are now open as to how many of these projects *E&CM* gets out before Sinclair!

The QL at a glance

Price	£399
Distribution	Mail order only
Processors	Motorola 68008, Intel 8049
Keyboard	Full travel, 65 Keys
Memory	128K RAM expandable to 640K 32K ROM expandable to 64K
Storage	2 Microdrives each with 100K storage, 3.5sec. av. access time
Expansion	Microdrive, RAM, ROM, 2 x LAN, 2 x RS232C, 2 x joystick. RGB and UHF sockets.
Graphics	Hi-res 4-colour 512 x 256 Lo-res 8 colour 256 x 256
Operating system	QDOS multi-tasking single-user system
Language	SuperBASIC
Software	Wordprocessor, spreadsheet, database, graphics

"Super BASIC puts right all those things which are wrong with BASIC".

Operating system

With characteristic Sinclair modesty, this is called QDOS. QDOS is a multi-tasking, single-user operating system which uses SuperBASIC (see below) as its command language. Again information is scarce: the section on QDOS in the manual is not yet complete. According to Sinclair, 'No existing operating system was remotely capable of capitalising on the full power of the 32-bit processor (Microsoft et al may have something to say about that) and that 'QDOS sets a new standard in operating systems for the 68000 family'.

Obviously multi-tasking is the most significant feature of QDOS. At the demonstration held when QL was launched, we saw an impressive display in which some four programs were run simultaneously in four windows, each coloured differently. Sinclair claim that up to 10 programs can be run simultaneoulsy on the screen, each program scrolled in any four directions, independently within its window.

QDOS has fully device independent input/output. A program can be written without reference to the type of device to be used, which can be specified when the program is run; and the QL can sense the type of peripheral devices connected to its expansion port and organise their input/output.

SuperBASIC

'Puts right all those things which are wrong with BASIC', according to Clive. A more detailed study of the manual following the press launch reveals that this is not such a far fetched statement. SuperBASIC adopts the structures and strategies of higher level languages such as BCPL and C.

The most notable aspect of the language is that it has taken the best parts of Spectrum BASIC and improved and enhanced them. Spectrum BASIC is an individual language, with some serious drawbacks but widely acclaimed for, for example, its string handling capabilities. The main disadvantage was of course its speed, or lack of it. SuperBASIC, as mentioned above, is used as the command language for QDOS, and this would imply that it is no slowcoach. One should expect nothing less from a language running on a 68000 based machine, but the final verdict must wait until full bench testing.

A few examples will suffice to illustrate the improvements that Sinclair has made. Number one: no more GOTO and GOSUB statements. GOTO is still supported by the language for compatibility with other

BASICs, however, the control structures available in SuperBASIC make the statement redundant. Looping in SuperBASIC is controlled by two basic program constructs: REPEAT/END REPEAT, and FOR/END FOR. These are used in conjunction with NEXT and EXIT. Processing a NEXT statement will either pass control to the statement following the appropriate FOR or REPeat statement, or if a FOR range has been exhausted to the statement following the NEXT.

Processing an EXIT will pass control to the statement after the END FOR or END REPeat selected by the identifier after the EXIT keyword. If an EXIT is used in a loop the loop must be terminated by END FOR or END REPeat. EXIT can be used to exit through many levels of nested repeat structures.

Another very useful command is SELect, which allows various courses of action to be taken depending on the value of a variable. The SELect variable is the last item on the logical line. Progressive statements are terminated by the next ON statement or by END SELect. Below is a sample program for the diagnosis of filing errors which illustrates the use of the SELect statement:

Listing: The SELect statement.

```
 10 SELect ON error_number
 20  ON error_number = 1
 30     PRINT "Divide by zero"
 40     LET error_number = 0
 40  ON error_number = 2
 50     PRINT "File not found"
 60     LET error_number = 0
 70  ON error_number = 3 TO 5
 70     PRINT "Microdrive file not found"
 80     LET error_number = 0
 90  ON error_number = REMAINDER
 90     PRINT "Unkown error"
110     error_recovery
120 END SELect
```

Software

Sinclair is onto a winner with an excellent suite of four software packs. It could be argued that the software itself is worth the £399 price tag (in which it is included).

The four programs were designed by Psion Ltd., and full advantage is taken of the machine's capabilities. Included is QL Quill, a word processor; QL Abacus, a spreadsheet; QL Archive, a database; and QL Easel, a graphics package. Each program is completely interactive: the results of commands are displayed instantly – what is seen on the screen is what will be dumped to the printer. A series of prompts is displayed in the control area at the top of the screen, and each pack is equipped with a HELP function. The programs are consistent in design and presentation so it is a simple matter to learn one and then another. All use colour, have a complete menu of options, use no complex equations, and the real beauty: data can be transferred from one to another.

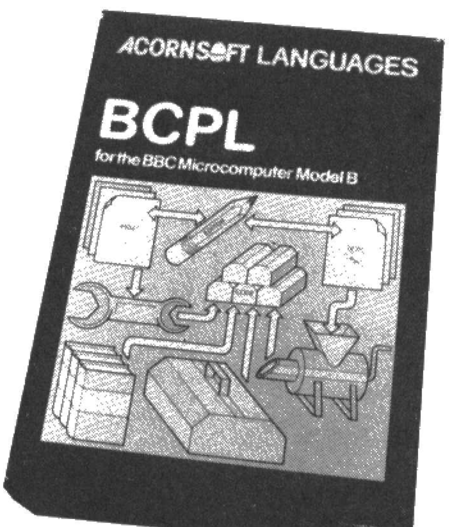
A demonstration was given of the Quill, and this package proved to be superior to many common business word processors, Wordstar to name but one. Few editing functions required the depression of more than one or two keys. Justification, margins, tabulation and insertion were immediate. Instructions could be obtained from the control window as to the operation of every aspect of the system: no constant recourse to the manual was necessary. Brief demonstrations of the other packages seemed to point to equal 'user friendliness' (disgusting expression). These three will not perhaps prove as useful to many as Quill, but will no doubt be a boon to many small businessmen who cannot afford £1500-2000 for what used to pass as a low-cost system.

Last word

Sinclair have opened a whole new market for computers by marketing the QL at such a low price. It will be an ideal machine for Estate Agents, shopkeepers, colleges, schools, hobbyists and who knows who else. This review can in no way be conclusive without a hands-on test of the machine. A full benchtest will appear in the next issue of *E&CM,* and we can expect to see a few niggling complaints, but hopefully no howlers by the Sinclair design team. ■

BCPL ON THE BBC MICRO

BCPL has been available from Acornsoft for two months, but few people appear to appreciate its potential as a systems programming language. Adam Denning has set out to change this state of affairs.



"BCPL offers much more than even BBC BASIC... if you can afford £99.65".

BCPL stands for *Basic Combined Programming Language,* and was developed by Martin Richards at Cambridge University in the late 60's. He was involved in the earlier CPL development, along with a few people from London University, but this project became too complicated, until eventually red tape and rivalry ended it. BCPL is specifically meant to be a simple but versatile language, ideal for writing compilers, interpreters and similar utilities. To this end, the BCPL compiler, the original BBC Basic, and the soon to be released Acornsoft version of LOGO are all written in BCPL.

Despite its obvious applications in this area, BCPL is also an exceptional learning language, being very easy to read and learn, and intrinsically well structured. In fact the better parts of BBC Basic such as procedures, indirection operators and repeat-until loops all come from BCPL. It hasn't caught on amongst the general public quite as much as it might mainly because some Americans, who at the time controlled the computer market completely, decided to alter it and call it C. This language is very well known, principally because it has nearly all of BCPL's virtues, and the Unix operating system is written in it. The might of America, or more specifically that of AT & T, the developers of Unix, ensured that Unix became very big very soon, hence C has overtaken BCPL. This is unfortunate, because a lot of programmers prefer the latter.

The Tripos operating system, written in BCPL, is far superior to Unix, but the developers just haven't got the sheer commercial weight required to market it properly. It is rumoured that Acornsoft's 16032 second processor will be supplied with Tripos.

The final word on this point goes to the authors of C:- "Many of the most important ideas of C stem from the older, but still quite vital, language BCPL . . . that language anticipated the vogue for structured programming by several years".

Acornsoft supply BCPL as a 16K ROM, along with a disc containing various utilities and a 450 page manual. The whole lot costs £99.65, and was supplied to Acornsoft by Richards Computer Products (RCP), owned by the brother of Martin Richards. The BBC implementation is very much disc-centered, and although it is possible to program in BCPL with only a cassette interface, this is definitely not recommended, and anyway Acornsoft refuse to supply a cassette version of the software. RCP probably will though.

The disc contains some essential programming aids, including the compiler, so tape-only users are advised to remember that the BCPL system will be virtually useless to them unless RCP or someone else comes to their aid. With the BCPL ROM in one of the BBC's sideways ROM sockets, typing *BCPL will invoke the language. A '!' prompt appears and certain built-in commands can be used, such as STORE, which tells you how the memory is organised, or SHUFFLE, which tidies up the store.

BCPL is especially notable for it's lack of *types.* While languages like Basic have integer variables, string variables, real variables, arrays and so on, BCPL simply has one data type – the word. With this word one can create vectors, which are equivalent to one-dimensional arrays, and that is it. Although this seems incredibly limited, it actually increases both the versatility and the simplicity of the language, as the user can make these words represent any conceptual data structure required. To quote Richards, "It might be, for instance, a time in milliseconds, a weight in grams, a function to transform feet per second into miles per hour, or it might be a data structure representing an employee record". In fact, the programmer has total control over what his data represents, and so he has to weigh the versatility in this against the responsivity.

The aforementioned vector is simply a consecutive group of words, the first element of which is pointed to by a variable. Using the ! indirection operator, any of these words can be accessed:

```
pointer!number := another number
```

this places the value 'another number' in element 'number' of the vector pointed to by 'pointer'. BCPL vectors unlike most Basic arrays start at element zero, so a

PROGRAM LISTING

```
 GET "LIBHDR"

//Version 1.3 A Denning 1/1/84
LET START () BE
 $(  LET item,file,hard = ?,?,?
     LET invec=VEC 7
     LET echo = FINDINPUT("/C")
     LET noecho = FINDINPUT("/K")
again:
     MODE (7)
     FOR I = 0 TO 1 DO
     WRITES("*X8D*X9D*X81SPOOLED FILE READER      (C) 1983 AD*N")
     WRITES("*N*N*X83File name:*X87")
     SELECTINPUT(echo)
     UNLESS RDITEM(invec,7) = 1 DO STOP(0)
     RDCH()
     file := FINDINPUT (invec)
     UNLESS file DO
      $( WRITES("*N This file does not exist. Please press  a*
         * key to start again.*N")
         SELECTINPUT(noecho)
         RDCH() ; GOTO again
      $)
     WRITES("*X86Do you want Paged Mode?*X87")
     SELECTINPUT(noecho)
     item := CAPCH(RDCH()) ='Y' -> TRUE,FALSE
     WRITES("*N*X85Do you want hard copy?*X87")
     hard := CAPCH(RDCH()) ='Y' -> TRUE,FALSE
     UNLESS MODE (3) DO STOP(15)
     IF item THEN WRCH(14)
     IF hard THEN WRCH(2)
     SELECTINPUT (file)
     item := RDBIN () & £X7F
  UNTIL item = ENDSTREAMCH DO
      $( IF (item >= 32 & item NE 127) | item='*N' THEN WRCH(item)
         item := RDBIN ()
         UNLESS item = ENDSTREAMCH DO item := item & £X7F
      $)
    ENDREAD ()
    WRCH(3)
    WRITES ("*N*NPRESS A KEY WHEN READY")
    SELECTINPUT(noecho)
    WHILE TESTFLAGS(CONSOLE.KEY) DO RDCH()
$)  RDCH () ; STOP(0)
```

To give a taste of the language, here is a short program to print out files onto the console.

The first line instructs the compiler to include the file LIBHDR in the source code. This file is the header file mentioned that declares the global procedures. The line starting // is equivalent to a REM statement in a Basic program, and is merely a comment. The procedure START is then declared. This procedure is declared in all BCPL programs, and serves to define the starting point. Three variables used in the program, *item*, *file* and *hard* are then declared, along with the vector *invec*, and the two input streams, *echo* and *noecho*. FINDINPUT("/C") opens a stream that works in a similar way to the one used when an INPUT statement is used in a Basic program, while FINDINPUT("/K") is similar to the GET stream.

We then have a label declaration, in order to use a GOTO (yes, even the most structured languages have them!). This label is *again*, and simply points the point in the program where it is declared. Then select Mode 7, print a title in double height, prompt for the name of the file to be printed, and select *echo* as the input stream. The RDITEM procedure then places the name typed in into the vector *invec*, and checks certain aspects of its validity. If an invalid filename was entered, the program stops. The variable *file* is then set to point to the stream control block created by FINDINPUT(invec), and if this file cannot be opened (ie if it doesn't exist), then a message to this effect is printed, and after a keypress the program jumps back to *again* and starts all over again.

Assuming a valid filename, the user is then asked if he wants the file to be displayed in the BBC's paged mode. The *noecho* input stream is selected for this question, and the rather neat:

item := CAPCH(RDCH()) = 'Y' → TRUE,FALSE

line reads the key pressed (by RDCH()), converts it to a capital letter with CAPCH(), and then compares it with the character 'Y'. If it was not Y, then *item* is given the value FALSE, while if it was Y, *item* becomes TRUE. The '→' is known as a conditional expression, and is rather more elegant than a group of IF . . THEN statements.

The same idea is then applied to the variable *hard*, this time asking if hard copy is required. Next, the file specified is selected as the source of input, Mode 3 is selected, and the file is printed out. When the end of the file is reached, RDBIN() returns the value ENDSTREAMCH, and this is detected and causes the program to end.

Although this was deliberately a simple program, it should be easy to see how much more flexible BCPL is than Basic.

declaration such as:

LET vec1 = VEC 4

will create a vector of five words, from vec1!0 to vec1!4.

On the BBC implementation, each BCPL word is 16 bits long, so there is no reason why the value contained in a word should not be considered as a 6502 address. Of course, such a consideration will only be valid if this is what the programmer intends. Also, as 16 bits is two bytes, it is obvious that each word can hold two ASCII characters. Indeed this is how strings are held in BCPL, so that a declaration such as:

LET string = "this is a string"

will create a vector pointed to by 'string', and each word of this vector will contain two consecutive characters of "this is a string". The first byte of the first word of a vector containing a string holds a number corresponding to the length of the string, so that, as in BBC Basic, no string can be over 255 characters long. When BCPL was first introduced, the compiler did not initialise strings in this way, and a procedure called PACKSTRING had to be used to move two characters into every word. The procedure UNPACKSTRING did the converse, and procedures GETBYTE and PUTBYTE were used to manipulate a packed string. Since then, the byte indirection operator, %, has been introduced, where

string%number

points to the number (n'th) byte in the vector. This means that

string%number + 256 *
string%(number+1)

and

string!(number/2)

are equivalent, (if, of course, number is even), and

string%O

is the length of the string if 'string' is a valid string.

BCPL has a whole host of standard procedures provided with it, most of these in the ROM. In order for a program to use these procedures, it must know where they are in the ROM, and this is where the *Global Vector* comes in. This is a large vector that all programs have access to, and a header file is included in the BCPL source code which declares a particular procedure as being GLOBAL n. Then, whenever that particular procedure is called, the program gets its address in the ROM from the n'th element of the global vector. Thus, communication between programs and sections of programs is made very easy. The programmer can declare some or all of his procedures or variables to be global, so that other parts of his program can access these easily.

One of BCPL's strongest points is the capability for input and output (I/O), with numerous procedures to open streams on all filing systems, including one special one of its own, called the *Store Filing System*. A

store file is treated identically to any other file, except that it is stored in the computer's memory along with the program. As a store file can be a program in its own right, this makes BCPL all the more powerful, as one can have a number of programs in memory all at the same time, any one of them being invoked whenever the user wishes.

Each file, when opened, is associated with a particular *stream*, so that common procedures can handle all sorts of I/O, rather than having particular procedures to deal with the keyboard, others to handle the disc, and so on. This is all done by small vectors called *stream control blocks*, which contain all the information for a particular open stream, such as the addresses of its get, put and close routines, its filename, and its operating system handle if it is a current filing system file.

So enhanced is the operating system under BCPL that the user immediately feels that he is using a real computer, and in fact a comparison with CP/M is very likely to severely embarrass Digital Research. Although most systems programming languages, including BCPL, are designed to be as portable as possible so that source code on one machine can be compiled on another with very little change, there are aspects of the BBC Micro that it would be foolish not to support, such as the VDU drivers, the sound, and the myriad interfaces. Acornsoft BCPL therefore provides procedures to handle all these, the VDU and sound procedures having identical syntax to their Basic counterparts, so that conversion from one language to the other is a simple process.

Having such advanced facilities is all very well, but how is a newcomer going to learn how to use it? The 450 page User Guide supplied with the system is uniformly excellent, but it does assume that you are familiar with the language. Beginners are strongly advised to buy the only good book on the language, *BCPL – The Language And Its Compiler* by Martin Richards and Colin Whitby-Strevens, published by the Cambridge University Press for £4.95. It is very readable, and an excellent introduction to the language. Acornsoft themselves are due to publish two books: a beginner's guide due out any time now, and an advanced guide, which I have just started writing.

Some programmers may feel that it is necessary to use floating point, and so RCP have released a floating point package, available either through them or Acornsoft for just under £20.00. With this package one has access to numbers of the same order that BBC Basic can handle, as well as some fast integer procedures, ideal for graphics. A very nice demonstration is provided with the package.

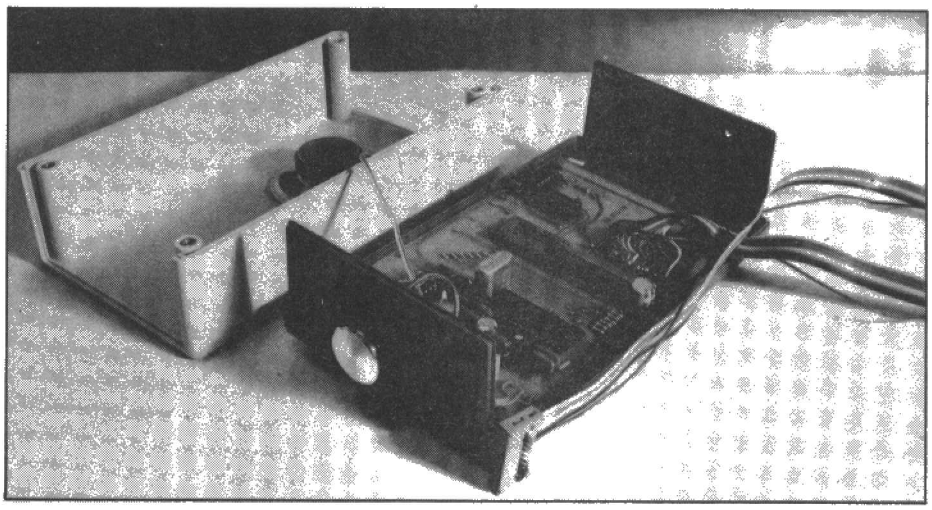
Further extensions are also available: a direct access package, containing seven procedures that allow the user access to any point in a disc file in a variety of ways. I can vouch for the quality of these procedures – I wrote them! There is also a stand-alone generator which allows conversion of BCPL programs into code that will run on any BBC Micro, regardless of whether it has a BCPL ROM installed.

As BCPL is a compiled language, programs are in the order of five times faster than their Basic counterparts, and the aforementioned global vector allows the creation of extremely large programs in a number of sections, joining them all together at the end. BBC BCPL is not truly compiled, as the ROM contains a *CINTCODE* interpreter, which runs the code that the compiler compiles into a very compact INTerpretive CODE, to allow large programs to fit into the machine. This intermediate stage does not slow the code down overmuch; a typical CINTCODE program is about half as fast as Acornsoft FORTH.

In conclusion, BCPL offers much more than even BBC Basic, and if you can afford £99.65 then this is a very useful place in which to spend it – BCPL has been described as the ideal hackers' language, as it can be as high level or as low level as you wish. For those that want that extra intimacy, there is also a fine assembler provided that allows the programmer to incorporate sections of 6502 machine code in BCPL programs as though they were actually written in BCPL. To aid in writing the programs, an excellent screen editor is provided, which also doubles as a very versatile word processor. ■

BBC speech synthesiser

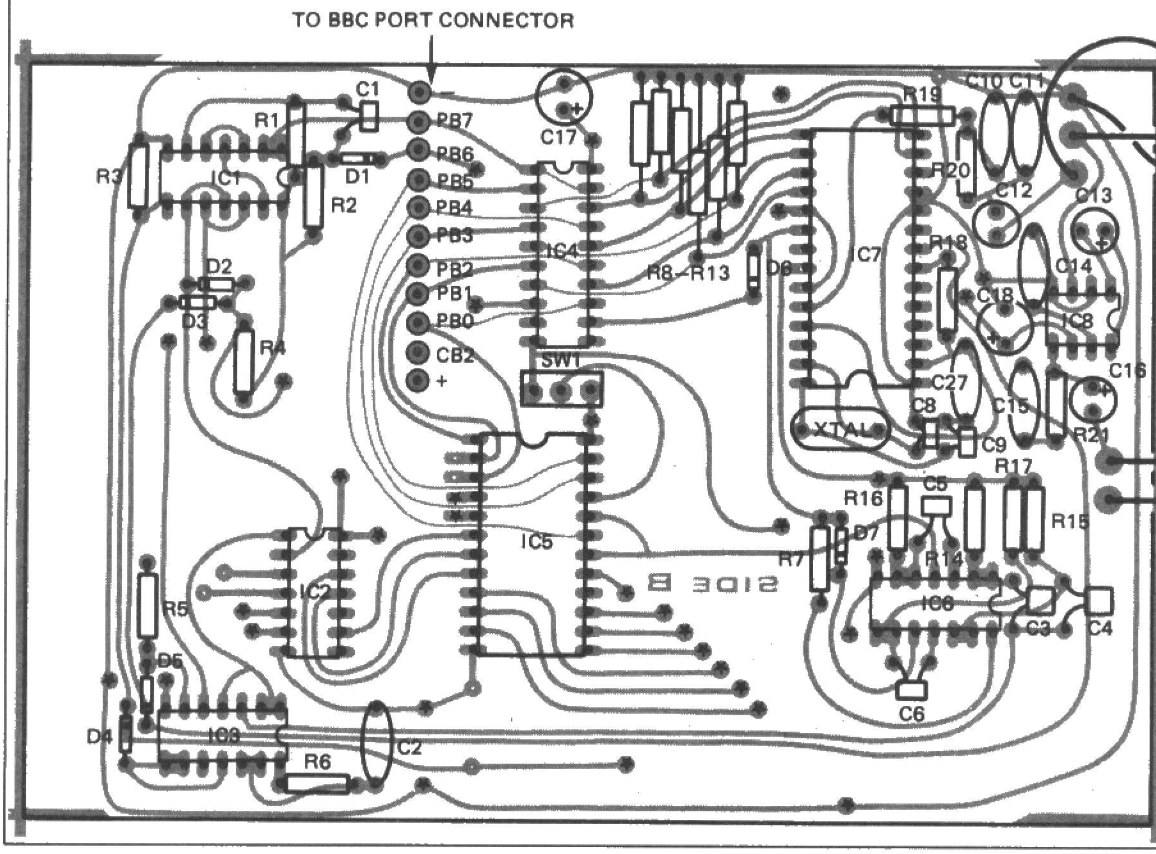
Last month Mark Stewart described the construction of a low cost speech synthesiser for the BBC micro. This month we publish the component overlay diagram to complete the project.



The magenta BBC speech synthesiser.

As explained last month, using the General Instruments allophone IC, the SPO256-AL2, this circuit allows two different modes of operation. In "direct" mode the computer controls the speech synthesiser IC directly, allowing total software control of individual allophones, which may be strung together to produce the required speech. In the second mode the computer selects a word or phrase from an on-board EPROM; 64 phrases, each up to 32 allophones long can be stored in a single 2716 2K EPROM. Alternative connections allow the format to be changed so that 32 phrases of 64 allophones, or 16 phrases of 128 allophones can be used.

It is also possible to use a 2732 4K EPROM and select between 2 banks of 2K using an on board switch. Switching between the two modes is done in software, so that a program can mix "standard phrases" from the EPROM with direct software controlled words.



A complete kit of parts for this project is available from Magenta Electronics. See their advert in this issue.

SPECTRUM TAP

Two tape recorders can be used at the same time using this Spectrum cassette controller designed by Robert Penfold.

The Sinclair ZX Spectrum has what is in certain respects one of the better cassette interfaces, giving reasonably fast saving and loading without compromising reliability. Its two principle drawbacks are a lack of any motor control and the fact that with most recorders it is necessary to keep connecting and disconnecting the "ear" and "mic" leads, since the interface will not operate properly while both are connected. This tape controller overcomes these two drawbacks, and, further, enables the Spectrum to be used with two recorders. Although primarily intended for use with dual cassette set-up, the unit can also be of benefit when using a single recorder.

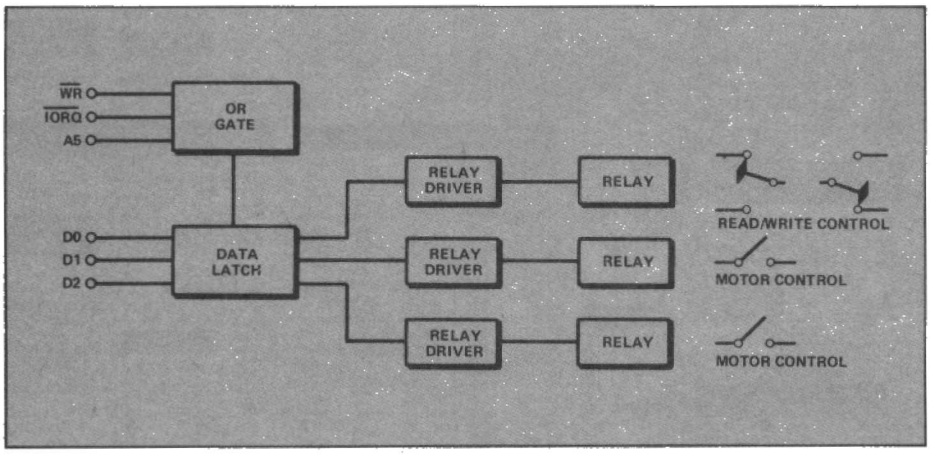


Figure 1. Block diagram of the dual tape control.

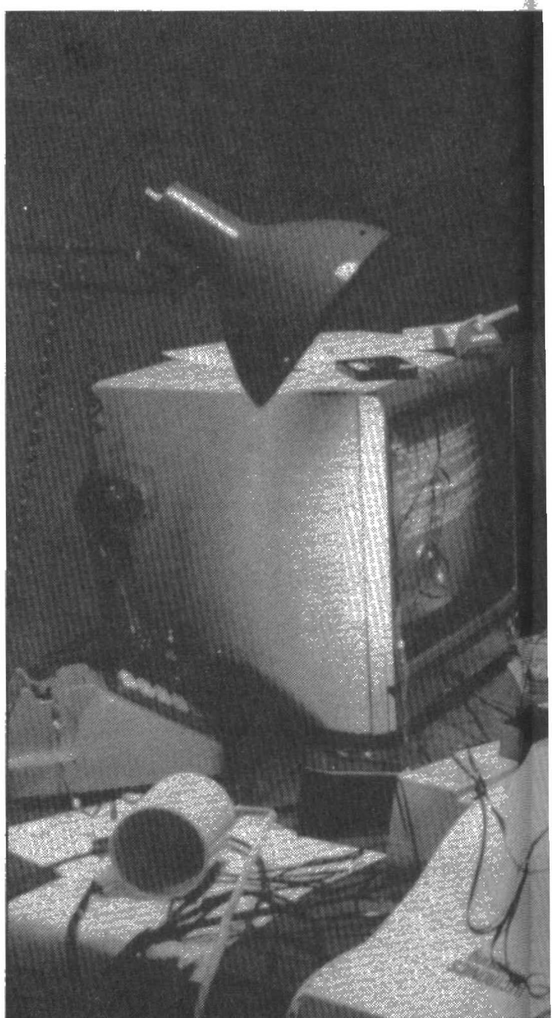
The purpose of the controller is to allow two tape recorders to be used at the same time; normally with one used for all READ operations and the other for all WRITE operations. Full motor control is provided under program control. The unit is of greatest use with data files where an existing file can be read in from one recorder, updated as required, and then recorded onto a fresh tape in the second recorder. By using this method a database can conveniently be recorded in sections, and therefore the total size of the records is not limited by the amount of RAM available. It does mean however, that the entire database cannot be sorted or searched in one go. A second advantage of a dual cassette system is that it eliminates a lot of cassette changing and rewinding, which can be irksome, and it also reduces the risk of a file being accidentally overwritten.

As the unit is basically just an output port driving three relays, it could obviously be adapted for use in any applications where a Spectrum must control equipment via one to three relays.

Operation

Two of the three relays are used to control the cassette motors, and the third controls which recorder is written to and which one is read from. The simple arrangement used in the controller is outlined in **Figure 1.**

Only three control lines of the Spectrum expansion port are used, write (WR), in/out request (IOREQ), and address line A5. The controller is placed in the I/O map of the Spectrum, and is operated using BASIC OUT instructions. The system of I/O control used in the Spectrum is to have all the address lines normally high, with one of the lower lines (A0 to A4) being taken low to activate an internal circuit such as the sound generator. The upper address lines (A8 to A15) are sometimes used to provide an internal I/O circuit with additional information. This leaves A5 to A7 completely free for use with external add-ons, and in this case A5 is used. The WR and IOREQ outputs go low when an OUT instruction is implemented, and address 65503 is used



so that A5 also goes low. This address leaves all the other address lines high so that no interference with internal input/output circuits occurs.

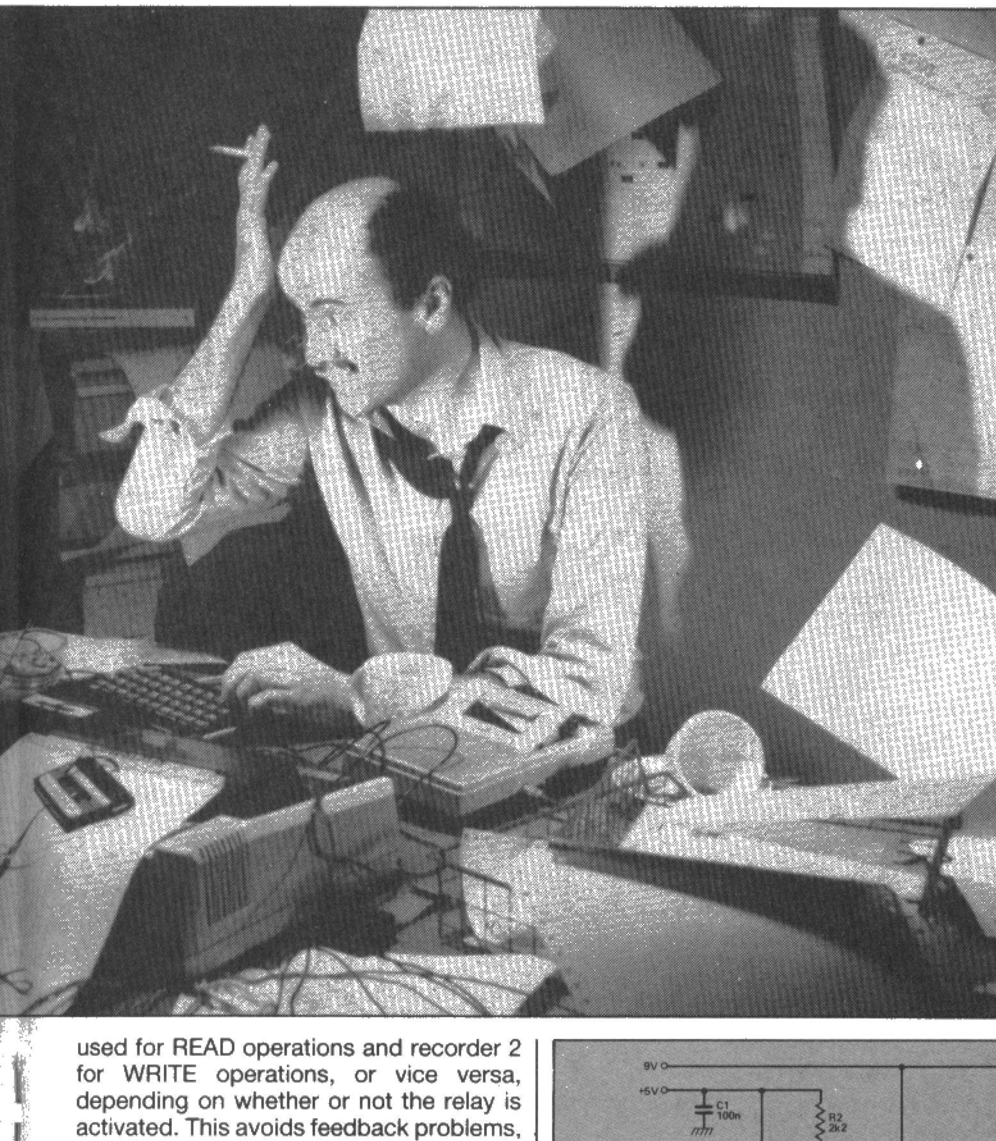
"the Spectrum lacks any motor control, and it is necessary to connect and disconnect the leads".

The negative output pulse produced by the OR gate during an OUT 65503,X instruction is used to operate a 3 bit data latch which is fed from the three least significant bits of data bus. Each output of the data latch drives a relay driver and relay. The relay controlled by data line D0 has double pole changeover contacts, and

COVER FEATURE

PE CONTROLLER

these are used to connect the "ear" and "mic" sockets of the Spectrum through to the two cassette recorders. However, at any one time only one recorder is connected to each socket, so that recorder 1 is



used for READ operations and recorder 2 for WRITE operations, or vice versa, depending on whether or not the relay is activated. This avoids feedback problems, and the need to manually disconnect any leads which are not actually being used.

The other two relays have straightforward SPST contacts, and are used to control the motors of the cassette recorders by way of the remote control sockets.

The circuit

Few components are used in the tape controller, as will be apparent from the full circuit diagram of **Figure 2.**

IC1 is the data latch, and this is a 74LS175 quad D type flip/flop. In this circuit only three flip/flops are required, and the fourth one is simply ignored. The latching pulse is supplied to the CP (clock pulse) input of IC1 by the 3 input OR gate formed by D1 to D3 plus R1. R2 and C2 are used to generate a negative pulse at switch on, and this is used to activate the master rest input of IC1. The point of this is merely to ensure that initially all three relays (and both cassette motors) are switched off.

The three relay drivers are identical, and each one consists of a current limiting base-feed resistor, an npn common emitter switching transistor, and a transient suppressor diode. A suppressor diode is needed in parallel with each relay coil to remove the high (reverse) voltage spike that is generated as this highly inductive component is switched off. This voltage could otherwise be sufficient to damage the semiconductor devices in the circuit. The diode provides a low impedance path so that the high impedance transient is clipped at a safe potential of only about 0.6 volts.

IC1 is powered from the regulated 5 volt output of the Spectrum, but the relays and driver circuits are powered from the unregulated 9 volt (nominal) supply. One reason for using this method is that it gives a greater choice of suitable relays, as most types are for 9/12 volt operation and will not operate from a 5 volt supply. It also helps to keep down the amount of current drawn through the Spectrum's 5 volt regulator, which is apt to run a bit on the hot side even without any external assistance. The 9 volt supply has a very high ripple content and is not stabilised, but a crude supply of this type is perfectly adequate for relay drivers.

Relays 1 and 2 are used to control the cassette motors, and note that output sockets SK1 and SK2 have neither output connected to earth. This is essential because most cassette recorders do not have either side of their "remote" sockets connected to earth, and will not function

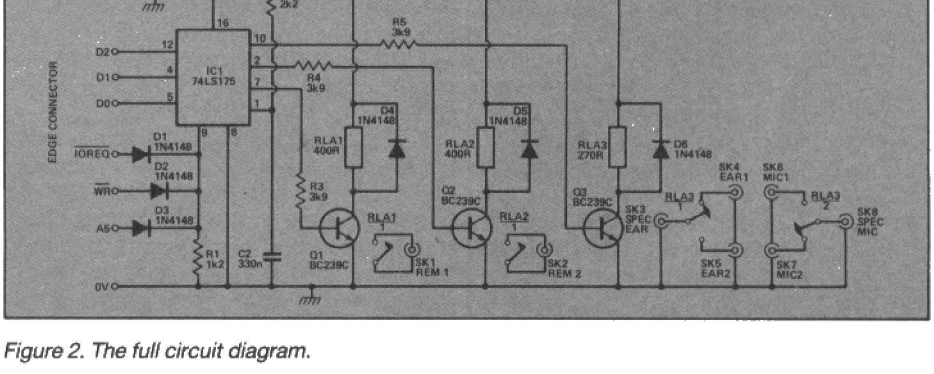


Figure 2. The full circuit diagram.

properly if either side should be so connected. Relay 3 is used to provide the read/write switching, and the way in which this operates is really self explanatory.

Construction

Construction starts with the printed circuit board, and this is illustrated in **Figure 3.** In most respects there is nothing out of the ordinary to note when constructing the board, but, as usual, care must be taken to connect the semiconductors the right way way round. Veropins are fitted at points where connections will eventually be made to the edge connector and sockets.

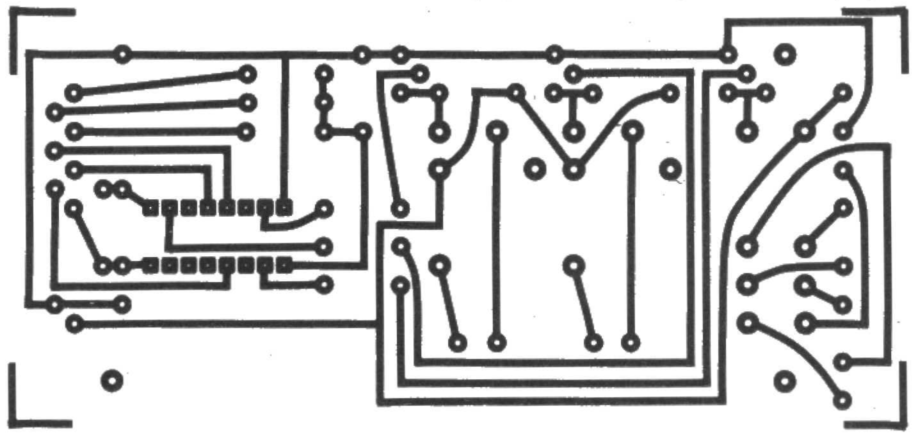
There is only one problem that could give any real difficulty, and that is the fitting of the relays. In order to mount the relays direct onto the board it is obviously necessary to use types having the correct pin spacing and layout, and this really means using the types specified in the components list as it is unlikely that any others will suit. From the electrical point of view, any relays having coils that will operate reliably from a supply of about 9 volts, a coil resistance of around 200 ohms or more, and suitable contacts, should be perfectly suitable. If alternatives are used though, either the printed circuit layout will have to be modified to suit, or a satisfactory way of mounting the relays and wiring them to the board must be found. Use of the specified relays is strongly recommended, especially for inexperienced constructors.

and these can then be connected to the Spectrum using the cassette lead supplied with the machine. The other sockets can also be jack types, with 2.5mm jacks used for SK1 plus SK2, and 3.5mm jacks for the rest. The controller would then be connected to the two cassette recorders using four 3.5mm and two 2.5mm jack leads. Mount the sockets in a sensible configuration so that connections to the Spectrum and recorders can be made easily and without confusion. An alternative method

"the unit could be put to use in any applications to control equipment via relays".

to earth. Either insulated (plastic body) sockets must be used for SK1 and SK2, or steps must be taken to insulate them from the case. If the unit is fitted in a plastic case a lead must be used to wire up the earth tags of the 3.5mm jacks, or the DIN sockets, as appropriate, but problems with insulating SK1 and SK2 are then avoided.

The connections from the printed circuit board to the Spectrum's expansion port are carried via a 9-way ribbon cable terminated in a Spectrum (2 x 28-way 0.1 inch pitch) edge connector. It is unlikely that 9-way ribbon cable will be available, but it is a simple matter to remove one lead from a piece of 10-way cable. Edge connectors for the Spectrum, complete with a polarising key, are now available from a number of component suppliers. The diagram on page 160 of the Spectrum manual gives



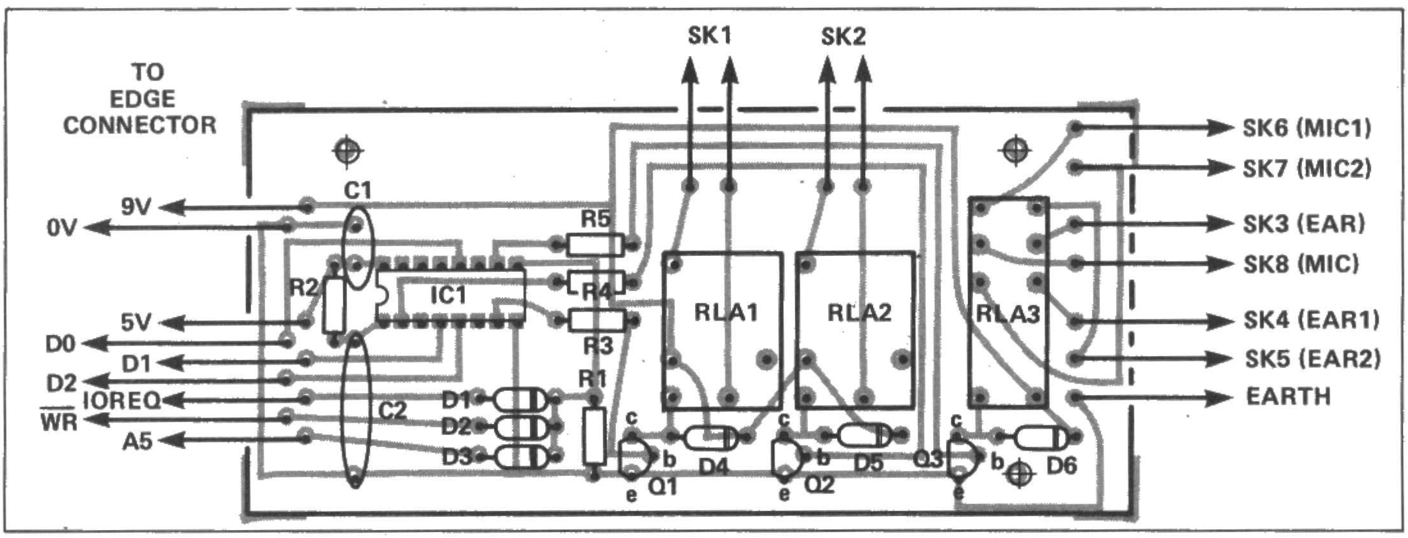


Figure 3. Foil layout and details of the printed circuit board.

The case for the prototype is an inexpensive aluminium type having approximate outside dimensions of 133 by 102 by 31mm, but any case which is not significantly smaller than this on any dimension should readily accommodate all the components. The printed circuit board is mounted on the base panel, leaving space for the sockets to be fitted on one of the case's end panels. 6BA or M3 fixings are suitable, and short spacers should be used over the mounting bolts to keep the connections on the underside of the board clear of the case.

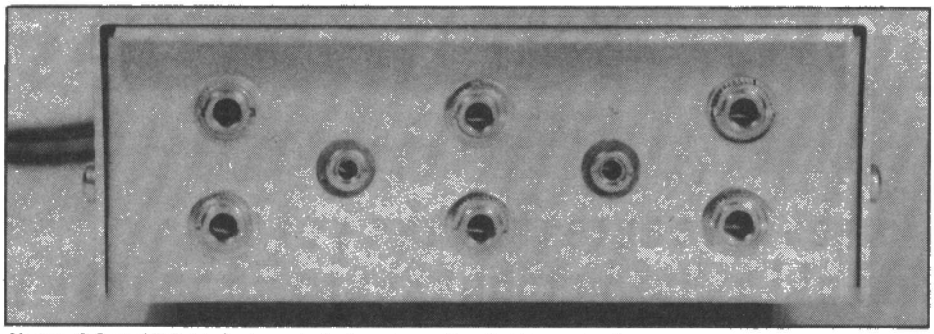
SK3 and SK8 are 3.5mm jack sockets, of making the connections to the recorders is to use standard DIN-to-DIN or DIN-to-jack computer cassette leads, and the controller would then need to be fitted with two 7 pin DIN sockets (to replace SK1 – 4 – 5, and SK2 – 6 – 7).

If jacks are used, together with a metal case, the case will carry the earth connection from one socket to the next, and the earth lead from the printed circuit board thus only needs to be connected to chassis tag of one of the jacks (it does not matter which one). A slight problem here is that the chassis tags of SK1 and SK2 connect to the appropriate relay contacts, and not

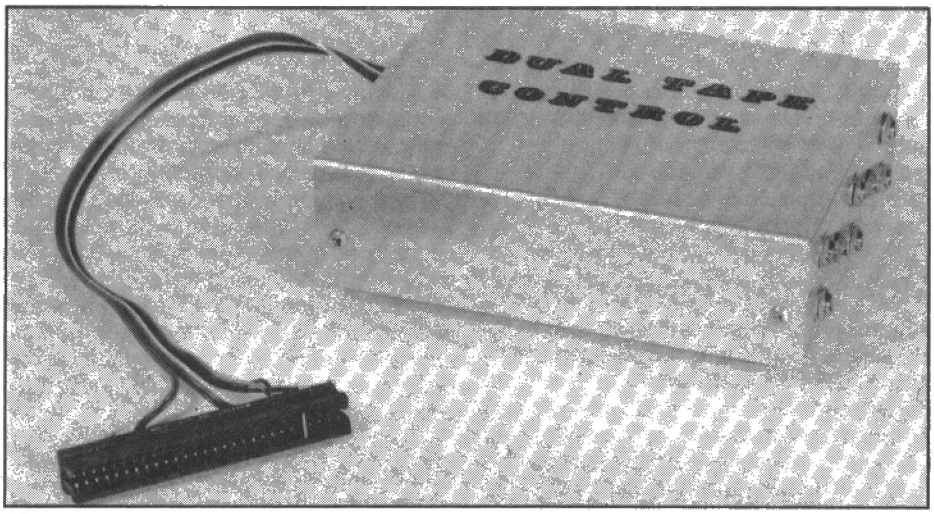
"... very useful with popular programs such as Vu-File".

details of the expansion port, but as this shows a "mirror image" of the port it can be a little confusing. **Figure 4** gives connection details for the edge connector, as viewed looking onto the pins at the rear of the connector.

With some cases it is possible to merely take the ribbon cable out through a gap between the two sections of the case, but



Above, 3.5 and 2.5mm jack sockets for connection to tape recorders.



Above, the completed unit and connector; below, inside the box.

an exit hole for the cable can be drilled at any suitable place in the box if necessary.

In use

Connect the unit to the expansion port of the Spectrum before applying power to the latter. When the Spectrum is switched on it should function normally, and none of the relays should switch on. As a quick check of the unit, try typing in these three commands:

```
OUT 65503,1
OUT 65503,3
OUT 65503,7
```

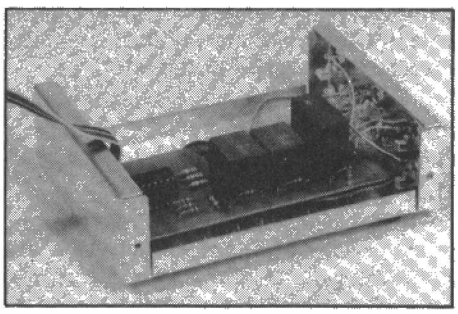
This should first switch on relay 3, then relay 2 as well, and then finally all three relays. OUT 65503,0 can be used to switch off all three relays.

In use the controller is not "user transparent", so LOAD, SAVE, and VERIFY instructions must be accompanied by the appropriate OUT commands. For recorder 1 the full instruction lines are as follows:

```
OUT 65503,4 : LOAD "program" :
  OUT 65503,0
OUT 65503,5 : SAVE "program" :
  OUT 65503,0
OUT 65503,4 : VERIFY "program" :
  OUT 65503,0
```

For recorder 2 they are:

```
OUT 65503,3 : LOAD "program" :
  OUT 65503,0
OUT 65503,2 : SAVE "program" :
  OUT 65503,0
OUT 65503,3 : VERIFY "program" :
  OUT 65503,0
```



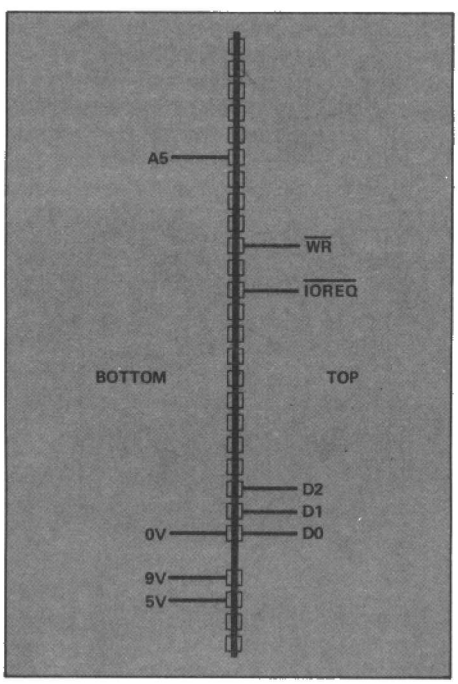


Figure 4. The edge connector.

To LOAD, SAVE, and VERIFY an array from within a program using two recorders the following would be used:

```
500  OUT 65503,3 : LOAD N$ DATA
     A$() : OUT 65503,0
700  OUT 65503,5 : SAVE N$ DATA
     A$() : OUT 65503,0
900  OUT 65503,4 : VERIFY N$ DATA
     A$() : OUT 65503,0
```

The array is called A$(), the filename is N$, and the line numbers are, of course, for example only.

The unit is very useful for use with a program such as the popular Vu-File from Psion, and such programs can be modified to work properly with the controller. For example, although Vu-File is mostly in machine code, it uses BASIC for input of filenames, plus SAVE, LOAD, and VERIFY operations on files. To break into the program answer "L" to the initial screen, enter anything as a filename, and when the program is trying to load press SHIFT/BREAK. You can then list the BASIC program. The relevant lines are 1005 (SAVE), 1100 (VERIFY), and 2000 (LOAD). They must be altered as follows:

```
1005  OUT 65503,5 : SAVE g$ CODE
      S,L : OUT 65503,0
1100  GO SUB 7000 : OUT 65503,4 :
      VERIFY g$ CODE : OUT 65503,0
2000  CLS : PRINT "Load a file" :
      GO SUB 6000 : GO SUB 7000 :
      OUT 65503,3 : LOAD g$
      CODE s : OUT 65503,0 :
      GO TO USRa
```

You should be aware that making a tape copy of the altered program may involve a breach of copyright. ■

PARTS LIST

Resistors (All 0.25 Watt 5%)

R1	1k2
R2	2k2
R3,4,5	3k9 (3 off)

Capacitors

C1	100nF ceramic
C2	330nF polyester

Semiconductors

IC1	74LS175
Tr1,2,3	BC239C (3 off)
D1 to D6	1N4148 (6 off)

Miscellaneous

SK1,2	2.5mm jacks (2 off)
SK3 to SK8	3.5mm jacks (6 off)
RLA1,2	400ohm 12 volt coil, SPCO contacts (Maplin 10A mains relay, order code YX97F) (2 off)
RLA3	270ohm 12 volt coil, DPCO contacts (Maplin 5A mains relay, order code YX98G)

Case about 133 x 102 x 38mm; Printed circuit board; 9 way ribbon cable and Spectrum (2 x 28 way 0.1 inch pitch) edge connector; 6BA fixings, wire, connecting cables, etc.

Micrographic Techniques

This month Mike James introduces the ideas of matrix transformations as an aid to the manipulation of graphics displays.

A simple two-dimensional shape is made up of straight lines that connect points. Last month's Micrographics introduced the idea that a good way to represent a simple shape inside a computer is by the dual system of a point file and a line file, the point file being a list of co-ordinate pairs specifying a number of points and the line file being a list of pairs of points that specify a number of lines. This hierarchical structure can be extended to include a number of lines making up a single shape and a number of shapes making up a single display. To do this we have to look at ways of changing the position and orientation of shapes so that they can be combined to make a complete display. For example, a landscape might use a number of tree shapes, identical apart from position, size and orientation and rather than store the line information for each it is better to keep one 'prototype' shape and generate all of the 'examples' of it using appropriate transformations.

Moving points

One of the additional advantages of storing the point and line data separately is that a shape can be transformed by applying the transformation to each of its points in turn and then drawing the straight lines that connect them. For example, if a square is defined by four points and the lines that connect them, a transformation that shifts the four points an equal amount to the left will be effective in shifting the entire square if the lines are redrawn between the same points but in their new position. This simple observation means that we need only involve the point file in transformations.

The next question is what sort of transformations on points are we interested in? If we restrict ourselves to transformations that preserve shape (on the grounds that shape is something defined before the transformation part of the graphics program) then the only transformations permitted (ie shape preserving) are:

translation
rotation
reflection
and scaling (change of size)

It just so happens that these transformations correspond to a subset of the transformations that can be implemented using matrices. Although matrices are not essential to the production or understanding of two-dimensional transformations they do make things a lot simpler when it comes to handling three dimensions (the subject of a later part of Micrographics).

Matrix transformations

A matrix is nothing more than a two-dimensional array with a special rule for multiplication. The two-dimensional array is familiar enough so the only possible source of trouble is the rule for multiplication. Although a point in two dimensions can be specified by two co-ordinates, for the purposes of matrix transformation there are advantages in using three numbers in the form of a 'column vector'. If the co-ordinates of the point are x,y then it is represented by the column vector:

$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

The reason for the apparently redundant final 1 is difficult to explain in detail but, put in simple terms, it is necessary to allow translations to be included in the matrix formulation. (In fact the use of three numbers to represent a point in two dimensions is a very useful mathematical technique called 'homogenous co-ordinates').

The result of a transformation is another column vector obtained by multiplying the original column vector by a 3 by 3 matrix. The multiplication is such that the first element of the new column vector is obtained by multiplying each element of the first row of the matrix with the corresponding element of the column vector and adding the results together. The second element of the new column vector is obtained by performing the same operation using the second row of the matrix and the third element is obtained using the third row. The operation of forming each element of the new column vector is often thought of as multiplying each row of the matrix by the column vector. (See **Figure 1**). As a BASIC program this gives:

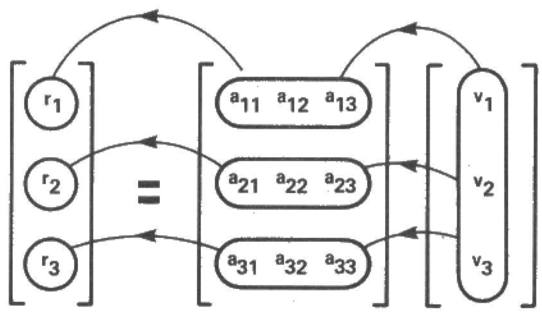
```
FOR I=1 TO 3
FOR J=1 TO 3
A(I)=T(I,J)*V(J)
NEXT J
NEXT I
```

assuming that the initial co-ordinates are stored in V, the result in A and the transformation matrix is T.

To convert a pair of co-ordinates into a column vector all that you have to do is add the final 1. However converting a column vector back to a pair of co-ordinates is not always as simple as ignoring the final 1. The reason for this is that during matrix multiplication it is possible for the final 1 to be changed into some other value. In other words the general form of a column vector is:

$$\begin{vmatrix} x \\ y \\ w \end{vmatrix}$$

where w might not be 1. To convert this general form of column vector into a pair of



$r_1 = a_{11} v_1 + a_{12} v_2 + a_{13} v_3$

r_1 IS ROW 1 'TIMES' COLUMN
r_2 IS ROW 2 'TIMES' COLUMN
r_3 IS ROW 3 'TIMES' COLUMN

Figure 1. The method of matrix multiplication.

co-ordinates it is necessary first to divide each element by w, giving:

$$\begin{vmatrix} x/w \\ y/w \\ 1 \end{vmatrix}$$

The final pair of co-ordinates are then simply x/w and y/w. Transformations that change the value of the final 1 are very important in three-dimensional graphics but in two-dimensional graphics they can be avoided and the co-ordinates can be recovered by simply throwing away the last element of the column vector.

Some transformations

So far all this talk of matrices is a little abstract and a few examples are long overdue. The matrix R:

$$\begin{vmatrix} \cos A & -\sin A & 0 \\ \sin A & \cos A & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

produces an anti-clockwise rotation about the origin through an angle A. That is the point given by RV is the same distance from the origin as the point V but moved through an angle A anti-clockwise. If every point in the point file is multiplied by this matrix and then the lines in the line file redrawn the resulting shape will be the same but rotated through an angle A about the origin.

The matrix M:

$$\begin{vmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{vmatrix}$$

produces a translation by tx units horizontally and ty units vertically. If every point in the point file is multiplied by S then the final shape will be removed by tx units horizontally and ty units vertically.

The matrix S:

$$\begin{vmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

produces a scale change such that all the x co-ordinates are enlarged by a factor sx and the y co-ordinates by sy. This matrix applied to each point in the point file will produce a scale change in the shape. This scale change is not quite as simple as an enlargement or a reduction in the overall size of the shape. Firstly, to ensure that the shape remains the same after the transformation, sx and sy must be the same. If they are different the shape will be distorted. Another complication is the role of the origin. In the same way that the first matrix produced a rotation about the origin, S produces a scale change centred on the origin. To understand what this means you first have to notice that the point at the origin ie 0,0 is the only point left unaltered by the scaling transformation.

Combining transformations

The transformation matrices given above are very useful but normally the required transformation is a combination of rotation, translation and scaling. For example, a square in the middle of the screen can be rotated about one of its corners by the following three steps:

1) translating the desired corner to the origin
2) performing the rotation about the origin
3) performing the translation in step 1 in reverse

(ie —tx,—ty)

The problem is that multiplying a point by a transformation matrix takes time and repeating the operation on all the points in the point file takes even longer. If you are going to implement a transformation using three or more different matrices then the total time spent in number crunching is certainly going to be unacceptable. The solution is that any number of transformation matrices can be multiplied together to give a single matrix with the same effect.

Multiplying two matrices together is a fairly simple operation. If A and B are two 3 by 3 matrices then the result of multiplying them together is another 3 by 3 matrix C with elements given by:

```
FOR I=1 TO 3
FOR J=1 TO 3
FOR K=1 TO 3
C(I,J)=A(I,K)*B(K,J)
NEXT K
NEXT J
NEXT I
```

This short program may look complicated but the best way to think of it is that element C(I,J) of the result is obtained by multiplying row I of the first matrix by column J of the second.

"... matrices make things a lot simpler when it comes to handling three dimensions ..."

The transformation that produces a rotation about any given point is such an important one that it is worth multiplying out the three matrices representing the transformations given at the start of this section. That is:

G = T1*R*T2

where G is the matrix that produces a rotation about the point x,y through the angle 'a' and T2 is a translation from x,y to the origin, R is a rotation about the origin through an angle 'a' and T1 is a translation from the origin back to x,y. Writing these matrices out in full gives:

$$\begin{vmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{vmatrix} \quad \begin{vmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad \begin{vmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{vmatrix}$$

and multiplying them together produces the result:

$$\begin{vmatrix} \cos a & -\sin a & -x\cos a + y\sin a + x \\ \sin a & \cos a & -x\sin a - y\cos a + y \\ 0 & 0 & 1 \end{vmatrix}$$

for the matrix G.

Experimenting with transformations

The power of matrix transformation is the range of transformations that can be encompassed in a single method. As a way of trying out the effect of any transformation matrix **Program 1** (written in Acorn BASIC) will apply it to a small square and plot the results.

For those of you interested in extending this program or converting it to other dialects of BASIC, the subroutine structure is:

subroutine	function
1000	set arrays X and Y to co-ordinates of four corners of square
1500	sets transformation matrix A to the identity
2000	draws co-ordinates in X and Y
3000	applies transformation in A to X and Y
4000	gets new elements of A from keyboard
5000	prints A

PROGRAM 1

```
10 MODE 4
20 DIM X(4),Y(4),A(3,3)
30 GOSUB 1000:GOSUB 1500
40 GOSUB 2000
50 GOSUB 4000
60 GOSUB 3000
70 GOSUB 2000
80 GOSUB 1000
90 GOTO 50

1000 X(1)=500:Y(1)=500
1010 X(2)=600:Y(2)=500
1020 X(3)=600:Y(3)=600
1030 X(4)=500:Y(4)=600
1040 RETURN

1500 FOR I=1 TO 3
1510 A(I,I)=1
1520 NEXT I
1530 RETURN

2000 MOVE X(1),Y(1)
2010 FOR I=2 TO 4
2020 DRAW X(I),Y(I)
2030 NEXT I
2040 DRAW X(1),Y(1)
2050 RETURN

3000 FOR P=1 TO 4
3010 T=X(P)*A(1,1)+Y(P)*A(1,2)+A(1,3)
3020 Y(P)=X(P)*A(2,1)+Y(P)*A(2,2)+A(2,3)
3030 X(P)=T
3040 NEXT P
3050 RETURN

4000 GOSUB 5000
4010 PRINT "A(";
4020 INPUT I
4030 PRINT TAB(10,30);",";
4040 INPUT J
4050 PRINT TAB(15,30);")=";
4060 INPUT A(I,J)
4070 GOSUB 5000
4080 RETURN

5000 PRINT TAB(0,26);
5010 FOR I=1 TO 3
5020 FOR J=1 TO 3
5030 PRINT TAB((J-1)*5);A(I,J);
5040 NEXT J
5050 PRINT
5060 NEXT I
5070 PRINT
5080 RETURN
```

At the start of the program the transformation matrix is set to:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

"... practical programs may have to abandon the matrix formulation to achieve reasonable processing speed ... however matrix algebra is the best way of manipulating transformations ..."

which is called the 'identity' matrix. When a column vector is multiplied by the identity matrix no change is produced and in this sense the identity matrix plays the role of 1 in matrix multiplication. After the matrix A has been printed out the program pauses and allows the user to type in a new value for any of the elements. As each element is altered the result of applying the transformation to the square is displayed. You might like to try altering the diagonal elements of A to show the effect of scaling. For example, if A(1,1) is changed to 2 the square will be changed to a rectangle that is twice as wide as high. What you might not have expected is that the rectangle is now twice as far from the origin as it was before the scaling. This can be corrected by changing A(1,3) to –500 to produce a translation back to the original position. You also might like to try:

$$\begin{vmatrix} 1 & 1.5 & -500 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

which is one of the many non-shape preserving transformations called a 'shear'.

Transformations in traditional graphics

In traditional graphics programs transformations are usually implemented by a general transformation routine. The user specifies a transformation matrix and the routine applies it. The transformation matrix is itself built up by a list of calls to a range of subroutines that perform individual transformations. Starting from an identity matrix each call modifies the current transformation matrix to give a final matrix that produces the total transformation. This is then applied to each point in the point file and the display drawn.

This traditional approach is not really appropriate for interactive graphics on a microcomputer. The reason for this is that the parameters of the transformation are best supplied from an input device such as the keyboard and then immediately applied to the point file. For example, rather than build up the 'rotation' about any point matrix by multiplying other more fundamental transformation matrices it is better to supply a 'rotate about any point' option within a graphics package. Even within this option it is better to avoid the matrix representation of the transformation in favour of explicitly writing out the equations that relate the new co-ordinates to the old. In other words, rather than using a general subroutine to multiply the column vector that represents each point in the point file but a 3 by 3 array that holds the co-efficients of the 'rotate about any point' matrix, it is better to use:

XN=(X(I)—X)*COS(A)+(Y—Y(I))*SIN(A)+X
YN=(X(I)—X)*SIN(A)+(Y(I)—Y(*COS(A)+Y

which give XN,YN the co-ordinates of the transformed point, in terms of X(I),Y(I) its original co-ordinates, X,Y the co-ordinates of the centre of the rotation and A the angle through which everything is rotated. These two equations are easily obtained by multiplying the transformation matrix G (given earlier) by the column vector:

$$\begin{vmatrix} X(I) \\ Y(I) \\ 1 \end{vmatrix}$$

and separating out the expressions for XN and YN.

At this stage it looks as though the effort that has been expended on studying the matrix formulation of transformations of shapes has been a waste of time in that the final program is better implemented using transformation equations. However, it should be clear that matrix algebra is an excellent way of describing and manipulating transformations. In general transformations are best described, thought about, discussed etc in terms of matrices but practical graphics programs often have to abandon the matrix formulation to achieve a reasonable speed of processing.

A rotation module of the line and point editor

As an example of the way that transformations can be used in interactive graphics a 'rotate about any point' module is a natural extension to the point and line editor given last month. The only modification necessary to the original program is to the command menu:

```
4050  IF A$="R" THEN GOSUB 8500
4060  RETURN
```

Following this modification pressing "R" will call the rotation module:

```
8500 PRINT TAB(0,24);"ANGLE ";
8510 INPUT A
8520 PRINT "SELECT CENTRE OF ROTATION"
8530 PRINT "PRESS ANY KEY WHEN READY"
8540 GOSUB 2000 '
8550 XG=X:YG=Y:GOSUB 3000:GOSUB 3000
8560 IF INKEY$(0)="" THEN GOTO 8540
8570 PRINT TAB(0,24);SPC(20)
8580 PRINT SPC(25):PRINT SPC(25)
8590 A=RAD(A)
8600 FOR I=1 TO P
8610 XT=(X(I)-XG)*COS(A)+(YG-Y(I))*SIN(A)+XG
8620 Y(I)=(X(I)-XG)*SIN(A)+(Y(I)-YG)*COS(A)+YG
8630 X(I)=XT
8640 NEXT I
8650 J=-1
8660 RETURN
```

This should be added to the point and line editor along with the modification to the command menu. The rotation module asks the user to specify the angle of rotation in degrees from the keyboard and the centre of rotation using the joystick. Once these two parameters have been entered the entire point file is subject to the transformation given by the pair of equations in the previous section. Following this, pressing "D" will cause the lines specified in the line file to be drawn, thus showing the newly rotated shape.

It is possible to add other modules to implement other transformations such as translation and scaling but these are much easier than rotation about any point and are left for your own experimentation.

More complications

So far the transformations that have been described have been applied to every point in the point file. This doesn't allow for the possibility that there are a number of different graphics 'objects' on the screen that should be transformed separately. This problem is taken up nex month along with the subject of global transformations.

Next Month – -Prototypes, objects, windowing and clipping.

Secret corrections

Sir,

I have to admit that a small error crept into my article "Secrets of the Spectrum's Streams and Channels" in the February issue of *E&CM*. Due to my ever increasing dyslexia (or rather its numerical equivalent) two numbers in the machine code were consistently reversed. In Program 1 the second and third numbers in line 10 should be 254,0 and the first line of Table 2 should read 01,254,0. In the same way the eighth and ninth numbers in line 10 of Program 2 should be reversed to read 245,0 and address 23301 of Table 3 should read 1,254,0. My mistake was consistently to place the high order byte of a 16-byte address first, when of course in Z80 assembly language the low order byte always comes first. Fortunately the error doesn't ruin the sense of the examples and they even work – although they send their data to more than one Spectrum I/O device at a time! It is also worth adding that the first example works even with the Microdrives and Interface 1 connected.

Please also note that line 1020 in Program 1 has been truncated and the final * 256 is missing. The full line should read:

```
1020  POKE c,23296–INT(23296/256)
      *256
```

With apologies,
Mike James

PRINTSPOOLER

What do you do while the printer is churning out hard copy? Chew your fingernails? This program by Chris Cant gets around the frustration of waiting, and allows you to continue writing while using any centronics linked printer.

This program gets around one of the most annoying problems in computing: doing some printing whilst still using the machine for other purposes. It will only work for centronics type printers on the parallel port; the *FX5 and *FX3 values are ignored. It has been used successfully using BBC BASIC and on other machines running Wordwise and View. First, the instructions!

Procedure

1 The hard bit. Type in **Listing 1.** Make sure that it is correctly entered, as a single wrong character could cause strange results. Save the resultant machine code as instructed in file 'print' in your disk program library. Also save files 'stop' and 'kick' to use in conjuction with the printer spooler. If you feel inspired make up your own names for these files.

2 Create a file on disk with the text which you wish print out. It may already exist in this form. Otherwise direct your word processor or other output to a disk file or SPOOL a basic listing.

BASIC:	*SPOOL listfl LIST or RUN to produce output *SPOOL
VIEW:	*SPOOL lisfl screen textfl *SPOOL

Note that VIEW will insert character code 14 to put the VDU in paged mode before screening the text. This may be interpreted by the printer as a special control code, eg. double width graphics on the Seikosha GP80A. Thus, the first thing the text contains should correct this, eg. by sending code 15 to a GP80A to put it in normal mode.

WORDWISE:	Option 8 should be used to spool output to a file

3 Enter the following command, where LIST is the name of the disk file which your output stored in:

*Print listfl

The program will send a carriage return to the printer and then carry on printing out text as obtained from disk. You will be able to carry out normal Basic editing and disk access as long as certain things are not done. The printer ignore character, as set by *FX6, will continue to be ignored. It is not a good idea to try a VDU2 command, although it might work for RS423 output. There will occasionally be slight pauses in your work as the system retrieves more data from disk. It should not lose any data from programs. The end of the printer spooler activity will be marked with a BEEP, as well as, no doubt, a quieter room.

4 If you wish to stop the spooler output before it has finished printing then the following command should be entered:

*stop

5 Occasionally it is possible that the spooler will stop before it has finished printing. This is usually caused by an unsuccessful call to one of the OS filing system routines and may be corrected for either by a 'good' OS filing system call or by entering the following command:

*kick

Program description

The operation of the printer spooler will be described in two stages corresponding to the listings 1 and 2. **Listing 2** (see next month) will be used to show how the basics work. It is written in non Tube compatible code and has not got some of the refinements of listing 1 but is less than 256 bytes long. The refinements on this mainly involve protecting this code from the user operations so that it is less likely to stop half way through. There are still pitfalls which the code cannot escape – a challenge to some readers.

Now you can begin entering the main listing and examining the ins and outs of the program. Next month *E&CM* will publish a shorter listing (see above) and a third short listing written for use with the spooler when used with the VIEW word processor. A full description will be given of all three programs, of procedures, interrupt routines, and enhancements. Now get programming!

LISTING 1 Printspooler

```
  10CLOSE#0
  20REM Printer Spooler    Chris Cant    August '83
  30
  40PROC_assemble
  45
  50PRINT''"Use:"'" *SAVE";CHR$(34);"Print";
    CHR$(34);~Q%;" ";~P%;" ";~Q%'
  60PRINT'" *SAVE";CHR$(34);"stop";CHR$(34);~eof;" ";~eof;" ";~eof'
  70PRINT'" *SAVE";CHR$(34);"kick";CHR$(34);~kick;" ";~kick;" ";~kick'
  80END
  90
1000DEF PROC_assemble
1010LOCAL I
1020
1030 OSBYTE=&FFF4
1040 OSBGET=&FFD7
1050 OSFIND=&FFCE
1060 OSWRCH=&FFE3
1080 OSARGS=&FFDA
1090 Posn  =0
1100 Q%=&1700
1110
1120 FOR I=0 TO 2 STEP 2
1130
1140 P%=Q%
1150[OPTI
1160.start  LDX  #&70        Get position of filename in command call
1170        LDA  #1
1180        LDY  #0
1190        JSR   OSARGS
1200
1210        LDX  &70
1220        LDY  &71
1230        LDA  #&40        Open file for READ
1235        JSR   OSFIND
1240        STA   chanel
1250        CMP  #0
1260        BEQ   fail
1270
1280        LDA  &204        Get old IRQ1V vector
1290        STA   IRQ1V
1300        LDA  &205
1310        STA   IRQ1V+1
1312
1314        LDA  #1          Allow printer disk activity
1316        STA   status
1318
1320
1325        SEI
1331        LDX  #11         Swap DFS vectors
1332.swap   LDA  &212,X
1333        STA   oldVEC,X
1334        LDA   newVEC,X
1335        STA  &212,X
1336        DEX
1337        BPL   swap
1338
1339
1340        LDA  #interr MOD 256  set up IRQ vector
1350        STA  &204
1360        LDA  #interr DIV 256
1370        STA  &205
1380        CLI
1390
1400        LDX  #0          Find printer ignore character
1410        LDY  #&FF
1420        LDA  #246
1430        JSR   OSBYTE
1440        STX   Pignor
1450
1460        LDA  #150        Clear old interrupts
1470        LDX  #&61
1480        JSR   OSBYTE
1490
1500        LDA  #150        Get old user PCR
1510        LDX  #&6C
1520        JSR   OSBYTE
1530
1540        TYA
1550        STA   oldPCR
1560        AND  #&F0
1570        ORA  #&0D        CA2 low; CA1 +ve edge
1580        STA   PCRlow
1590        ORA  #&0F        CA2 high
1600        STA   PCRhi
1610        TAY
1620
1630        LDA  #151        Write user PCR
1640        LDX  #&6C
1650        JSR   OSBYTE
1660
1670        LDA  #151        user DDRA as outputs
1680        LDX  #&63
1690        LDY  #&FF
1700        JSR   OSBYTE
1710
1720        LDA  #151        user IER enable
1730        LDX  #&6E
1740        LDY  #&82
1750        JSR   OSBYTE
1760
1770        LDY  #&D         Initial CR to printer
1780        LDA  #151
1790        LDX  #&61
1800        JSR   OSBYTE
1810        JSR   strobe
1820        RTS
1830
1840.fail   LDA  #7          File cannot be opened.
1850        JSR   OSWRCH
1860        RTS
1862
1864.kick   LDA  #1
1866        STA   status
1868        RTS
1870
1880
1900.status EQUB  0
1910.chanel EQUB  0
1920.Pignor EQUB  0
1930.IRQ1V  EQUW  0
1940.oldPCR EQUB  0
1950.PCRlow EQUB  0
1960.PCRhi  EQUB  0
2030
2040
2080.interr LDA  &FC         Save registers
2082        PHA
2084        TXA
2086        PHA
2090        TYA
2095        PHA
2100
2110        LDA  #150        Check interrupt type
2120        LDX  #&6D
2130        JSR   OSBYTE
2140        TYA
2150        AND  #2
2160        BNE   itsme
2170
2180        PLA              System interrupt
2185        TAY
2190        PLA
2195        TAX
2200        PLA
2210        JMP  (IRQ1V)
2220
2230.itsme  LDA  #150        Clear interrupt
2300        LDX  #&61
2310        JSR   OSBYTE
2320        CLI
2322
2324        LDA   status     Check no other disk activity taking place
2326        BNE   retry
2328        JMP   send         if so send out a dummy '0' to printer
2329
2330
2340.retry  LDY   chanel     Get next byte
2350        JSR   OSBGET
2360        BCS   eof
2370
2380        CMP   Pignor
2390        BEQ   retry
2400
2410.send   SEI
2420        TAY              Write byte to printer
2430        LDX  #&61
2440        LDA  #151
2450        JSR   OSBYTE
2460        JSR   strobe
2470
2480        PLA              Normal print return
2490        TAY
2500        PLA
2510        TAX
2520        PLA
2530        RTI
2540
2550
2552\       'EOF' found.  Close file. Stop printer
2554
2560.eof    LDA  #151        Reset user PCR
2570        LDX  #&6C
2580        LDY   oldPCR
2590        JSR   OSBYTE
2600
2610        LDA  #151        Disable CA1 interrupt
2620        LDX  #&6E
2630        LDY  #2
2640        JSR   OSBYTE
2650
2660        LDA  #0          Close file.
2670        LDY   chanel
2680        JSR   OSFIND
2690
2700        SEI
2710        LDA   IRQ1V      Reset IRQ1V vector
2720        STA  &204
2730        LDA   IRQ1V+1
2740        STA  &205
2741
2742        LDX  #11         Reset DFS vectors
2743.reswap LDA   oldVEC,X
2744        STA  &212,X
```

```
2745         DEX
2746         BPL   reswap
2747
2750         CLI
2760
2770         LDA   #7
2780         JSR   OSWRCH      Beep for finish
2790         PLA               Return for last time
2800         TAY
2810         PLA
2820         TAX
2830         PLA
2840         RTI
2850
2860
2870.strobe  LDA   #151        Waggle STROBE for printer
2880         LDX   #&6C
2890         LDY   PCRlow
2900         JSR   OSBYTE
2910
2920         LDA   #151        .. and return high
2930         LDX   #&6C
2940         LDY   PCRhi
2950         JSR   OSBYTE
2960         RTS
2970
3080
4000\
4010\  Each OS filing system vector is changed over to a new value.  This
4020\ allows the interrupting printer routine to know if there is any current
4030\ disk activity.  On entry to each routine the 'status' byte is set to
4040\ zero; on exit it goes to 1.  The printer routine will not attempt a
4050\ disk access unless 'status' is at 1.  Otherwise it will send a null
4060\ character, a zero, to the printer and try again when it is acknowledged.
4070\
4100.OSFScl  PHA : PHA             Save space on stack
4110         PHA : PHA
4115
4120         PHP : PHA             Save registers
4130         TXA : PHA
4140         TYA : PHA
4150
4160         TSX
4170         LDA   #(return-1) MOD 256  Insert address for OS FS RTS
4180         STA   &107,X
4190         LDA   #(return-1) DIV 256
4200         STA   &108,X
4210
4220         LDA   &109,X          Get call parameter
4222         LSRA
4224         TAY
4230
4240         LDA   posn,Y          Store OS FS address for RTI
4250         STA   &105,X
4260         LDA   posn+1,Y
4270         STA   &106,X
4280
4290         LDA   #0              Indicate start of FS
4300         STA   status
4310
4320         PLA : TAY             Restore registers
4330         PLA : TAX
4332         PLA
4335         RTI                   Jump to Filing System
4340
4350
4360.return  PHP : PHA             Save registers
4370         TXA : PHA
4380
4390         TSX
4400         LDA   &102,X          Re-adjust stack
4410         STA   &104,X
4420         LDA   &103,X
4430         STA   &105,X
4440
4450         LDA   #1              Indicate end of FS
4460         STA   status
4470
4480         PLA : TAX             Get registers
4490         PLA : PLA
4500         PLA : PLP
4510         RTS                   Final return
4520
4699
4999.oldVEC

5000.FILEV   EQUW  0               Old vectors stored here
5010.ARGSV   EQUW  0
5020.BGETV   EQUW  0
5030.BPUTV   EQUW  0
5040.GBPBV   EQUW  0
5050.FINDV   EQUW  0
5060
5090.osvec
5100.OSfile  JSR   OSFScl          New Filing system routines
5105         EQUB  0
5110.OSargs  JSR   OSFScl
5115         EQUB  0
5120.OSbget  JSR   OSFScl
5125         EQUB  0
5130.OSbput  JSR   OSFScl
5135         EQUB  0
5140.OSgbpb  JSR   OSFScl
5145         EQUB  0
5150.OSfd    JSR   OSFScl
5155
5160.OSfind  CMP   #0              Check against CLOSE#0 calls
5170         BNE   OSfd
5180         CPY   #0
5185         BNE   OSfd
5190         RTS
5195
5200.newVEC  EQUW  OSfile          New Filing system vectors
5210         EQUW  OSargs
5220         EQUW  OSbget
5230         EQUW  OSbput
5240         EQUW  OSgbpb
5250         EQUW  OSfind
5260
9000]
9005         posn=oldVEC-(((osvec+2) MOD 256) DIV 2)
9010NEXT
9020PRINT''~Q%;" -> ";~P%;"  (";P%-Q%;" bytes)."''
9030ENDPROC
```

Electron RS423

In Part 2 of the Electron communications interface project R. A. Penfold describes the control registers and software.

As mentioned in Part 1, there are four registers in the 6850. The simplest of these are at ?&FC01, and these are the transmit register (when writing) and the receive register (when reading). The other two are at ?&FC00, and are the status register (reading) and control register (writing).

If we consider the control register first, bits 0 and 1 control TX/RX clock division rate and master reset, as follows:

Number written (decimal)	Function
0	Divide by 1
1	Divide by 16
2	Divide by 64
3	Master reset

Number written (decimal)	Word format
0	7 bits, even parity, 2 stop bits
4	7 bits, odd parity, 2 stop bits
8	7 bits, even parity, 1 stop bit
12	7 bits, odd parity, 1 stop bit
16	8 bits, 2 stop bits
20	8 bits, 1 stop bit
24	8 bits, even parity, 1 stop bit
28	8 bits, odd parity, 1 stop bit

A master reset (?&FC00 = 3) should always be used before using the interface, otherwise it will almost certainly fail to operate at all.

The word format is controlled by bits 2, 3, and 4 of the control register, and the table given above shows the formats available, plus the required control number.

The RTS output and interrupts are controlled by bits 5 to 7 of the control register, but as interrupts are not available in this case, only control of the RTS output might be of importance. This is controlled as detailed below:

Number written (decimal)	Function
0	RTS = low
64	RTS = high

Of course, the number written to the control register is the sum of the numbers which give the desired functions and formats. For example, a baud rate of 300 (divide by 16), 8 bits plus 2 stop bits, and RTS = low, requires 17 to be written to ?&FC00 (1 plus 16 plus 0 = 17).

Bit 0 of the status register is perhaps the most important one as this goes high to indicate that fresh data is available from the receive register. It is automatically reset when the receive register is read. In practice this bit would be read and used to prevent the receive register from being read unless fresh data was available. The logic AND function together with a masking number of 1 can be used to read just bit 0 of the status register. These points are illustrated by line 40 of the test program of **Listing 1.**

Bit 1 of the status register has a similar function, and is used to indicate whether or not the transmit register is empty. When necessary, this bit can be read using the logic AND function plus a masking number of 2, and used to prevent data being set to the 6850 at an excessive rate.

Bit 3 of the control register reflects the state of the CTS input, and when high bit 1 of the control register (the transmit register empty bit) is inhibited. In most cases where the CTS handshake line is utilised it is therefore unnecessary to read bits 1 and 3 of the status register, and reading just bit 1 is sufficient.

Bits 4, 5, and 6 of the control register respectively indicate framing error, receiver overrun, and parity error.

Obviously the software for use with the interface will have to be designed to suit each particular application. The test program of listing 1 may well be sufficient for applications where the computer must operate as a simple data terminal where characters typed at the keyboard must be transmitted, and characters sent or "echoed" from the other data terminal must be displayed on the screen. The main **Listing 2** is for a similar type of program, but this demonstrates how language can be used to give a higher operating speed. Also, unlike the simple program of listing 1, this one provides a line-feed with each carriage return. The 6850 is quite straightforward to use, and it should not be too difficult to design software to suit a given application. ■

LISTING 1

```
>
   5 REM listing 1
  10 ?&FC00 = 3
  20 ?&FC00 = 21
  30 CLS
  40 IF (?&FC00 AND 1) = 1 THEN PRINT CHR$(?&FC01);
  50 A$ = INKEY$(0)
  60 IF A$<>"" ?&FC01 = ASCA$
  70 GOTO 40
```

LISTING 2

```
10OSBYTE=&FFF4:OSASCI=&FFE3
20FLAG=&FC00:PORT=&FC01
30 DIM SET 100
40 FOR I%=0 TO 3 STEP 3
50P%=SET
60[OPT I%
70.SET
80LDA #3
90STA FLAG
100LDA #21
110STA FLAG
120.CHECK
130LDA FLAG
140AND #1
150BEQ KEY
160LDA PORT
170JSR OSASCI
180.KEY
190LDX #0
200LDY #0
210LDA #129
220JSR OSBYTE
230CPY #&FF
240BEQ CHECK
250CPY #0
260BEQ SEND
270CPY #&1B
280BEQ ESC
290.SEND
300STX PORT
310CLC
320BCC CHECK
330.ESC
340BRK
350EQUS " Escape Pressed"
360BRK
370]
380NEXT I%
390 CLS:CALLSET
```

RAM in your ROM socket

For those without the facilities to build *E&CM's* fully featured Sideways ROM board, here is a new approach: inserting 2K of paged RAM in the BBC's sideways ROM area.

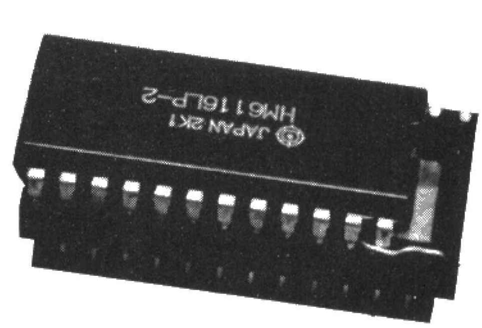
Most BBC owners have at least heard of their machine's ROM paging capability. This allows various pieces of software in ROM chips to reside in the same address space, only one actually being 'paged' in at any instant. It is not so widely known that it is also possible to fit random access memory (RAM) in this space. This can be used to develop paged ROM software in situ or to store data tables for use in BASIC or machine code programs. This article shows you how to fit this RAM and gives you the software to use it.

As always with high technology machines similar to the BBC micro, independent manufacturers rush to bring out peripherals to enhance and improve the basic micro's performance. Recently a plethora of ROM boards have appeared, all costing around £40. Most of these also offer the option of fitting CMOS RAM which you can use for the aforementioned development and data storage. What you are not told when buying these boards is that 8K of CMOS RAM costs about £40, the same price as the board itself. So 16K of CMOS RAM will cost you £80. This will often be outside the average BBC micro owners pocket.

2K CMOS RAM chips are available from various suppliers for £7. These are the Hitachi 6116 LP types. 2K of extra RAM is usually sufficient to develop paged ROM software (one utility at a time) and will allow 500 integer variables to be stored.

Fitting the RAM is not difficult but does involve connecting one wire to another chip, this does not have to be soldered (but soldering is advisable). **Figure 1** shows how this is done and you are advised to refer to this in the following explanation. Gaining access to the sockets is done as follows:

(i) switch off the machine and unplug it from the wall socket



The RAM chip in its 28-pin socket.

(ii) remove the four screws marked 'fix'

(iii) remove the two or three keyboard fixing screws, unplug the keyboard cable and unplug the wire to PL15. Place the keyboard in a safe place ready for reassembly.

The four sockets which interest us are located to the bottom right of the circuit board. If you have a standard machine and have not had any paged ROM's such as WORDWISE or VIEW fitted there will only be the BASIC chip plugged in. This will usually be in the socket fourth from the right hand side. Remove this chip with the aid of a flat bladed object, being very careful not to bend any pins. Plug this into the rightmost socket, again being careful not to bend any pins. Reconnect the keyboard and PL15 and switch the machine on. The usual screen message should appear. If it does not it means that your machine has not been upgraded to take paged ROM's.

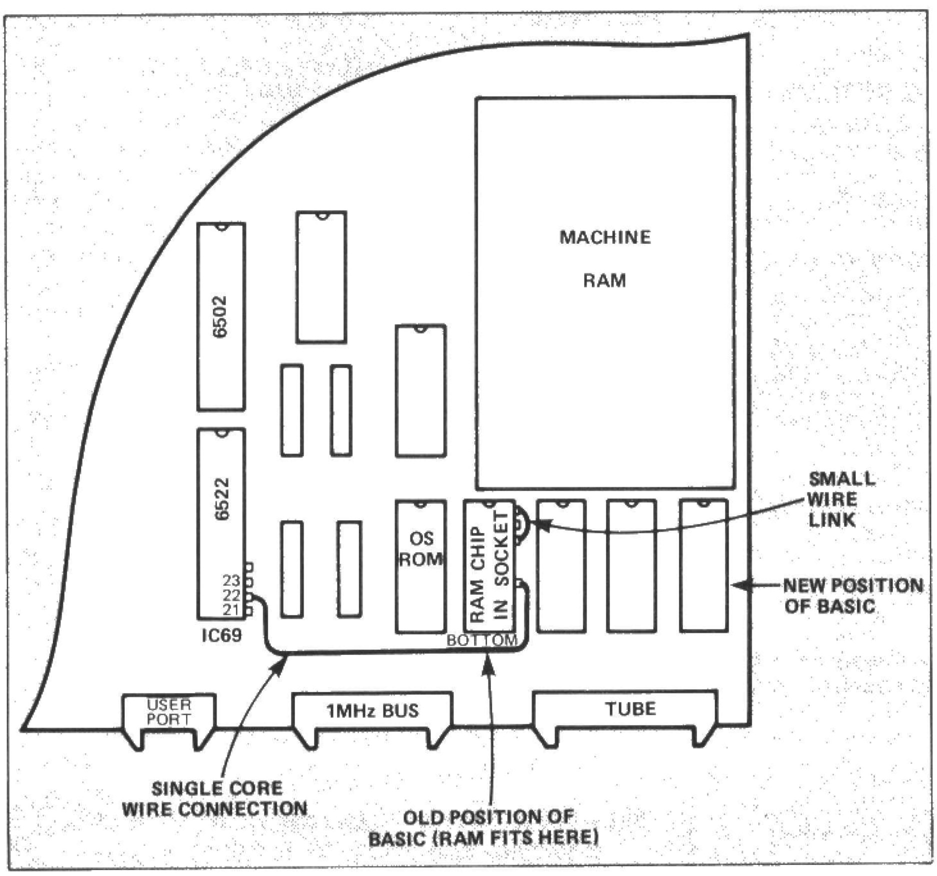


Figure 1. The RAM chip must be fitted in the 4th socket from the right.

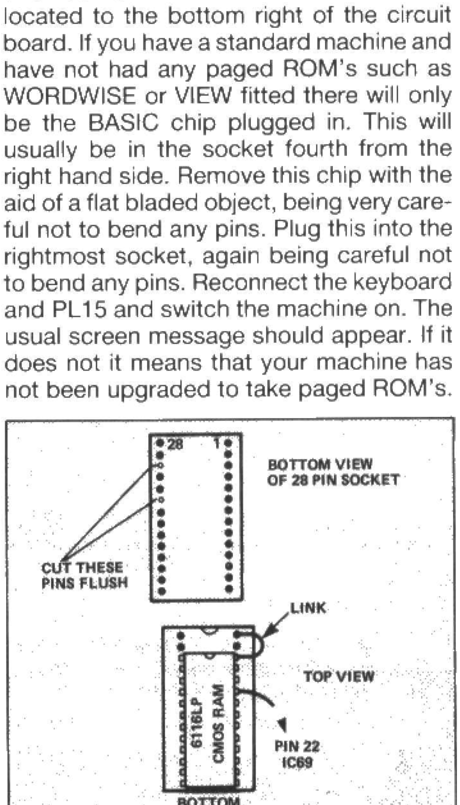


Figure 2. Showing modifications to the 28-pin socket.

The author recommends that this is done by your local dealer. If the usual message does not re-appear you will have to move the BASIC chip back to its original socket and re-assemble the machine. If your machine is fitted with a disk or econet interface some other ROM's will be present. It is recommended that all the chips are moved, if necessary, such that the BASIC chip is in the rightmost socket with the others in any other socket except the one fourth from the right hand edge.

Assuming that you did get the normal message on power up and that the socket four from the right is empty we can proceed.

CMOS RAM chips are very prone to destruction by static electricity so touch a metal fire or cold water tap before handling the chip. You will also need a 28 pin socket obtainable from most electronics stockists. Cut away the pins (Nos. 23 and 26) shown in **Figure 2** from this socket and plug in the CMOS RAM chip to the socket. Connect pin 24 of the chip to pin 28 of the socket using a small piece of insulated single core wire. Connect another piece of wire to pin 21 of the RAM chip long enough to reach pin 22 of the 6522 chip. The wires to the CMOS RAM chip will probably have to be soldered. Do this away from your BBC micro and NOT over the top of it or inside it. Plug the modified socket into the vacated socket on the circuit board. Finally push the flying end of the longer wire into pin 22 of the 6522 socket. As mentioned before, **Figure 1** clarifies this procedure. Re-assemble your machine and check that it operates as normal. If it does not check the installation procedure.

'2K of extra RAM is sufficient to develop paged ROM software'.

Software

After this is done we can consider the software necessary to write to and read from the RAM. **Listing 1** shows two procedures to do this one byte at a time.

To read from the RAM at address X, X being in the range &8000 to &87FF, call PROCread with PROCread(X). The byte is returned in the integer variable A%.

To write 'byte' to the RAM at address X, X being within the above range, call PROC-

LISTING 1

```
10    REM Program 1
20    REM RAM read and write
      procedures
30    REM for use with 6116LP
      CMOS RAM
40    REM Stephen A. Todd. (c) 1983
50
60    REM PROCmach must be
      executed once
70    REM before PROCwrite is used
80
90    PROCmach
100
110   END
120
130   DEF PROCmach
140   code%=&A00:REM or any safe
      place
150   P%=code%+4
160
       [                  OPT0
170                       LDA &70
180                       PHA
190                       LDA &71
200                       PHA
210                       LDA code%
220                       STA &70
230                       LDA code%+1
240                       STA &71
250                       LDY #0
260                       LDA #12
270                       STA &FE30
280                       LDA code%+2
290                       STA (&70),Y
300                       PLA
310                       STA &71
320                       PLA
330                       STA &70
340                       LDA #252
350                       LDX #0
360                       LDY #&FF
370                       JSR &FFF4
380                       STX &FE30
390                       RTS
400    ]
410   ENDPROC
420
430   DEF PROCread(address%)
440   ?&F6=address% MOD 256
450   ?&F7=address% DIV 256
460   Y%=12
470   A%=USR&FFB9 AND &FF
480   ENDPROC
490
500   DEF PROCwrite(address%,byte%)
510   !code%=address%:?
      (code%+2)=byte%
520   CALL(code%+4)
530   ENDPROC
```

LISTING 2

```
10    REM Program 2
20    REM CMOS 6116LP RAM checker
30    REM This program checks that
40    REM the RAM is functioning
50    REM Stephen A. Todd. (c) 1983
60
70    PROCmach
80
90    MODE6
100   VDU23;8202;0;0;0;
110   PRINT"RAM checker."
120   PRINTTAB(21)"Address"
130   PRINT"RAM write 1"
140   FOR T%=&8000 TO &87FF
150   VDU31,17,VPOS-1:PRINT~T%
160   PROCwrite(T%,T% MOD 256)
170   NEXT
180   PRINT"RAM read check 1"
190   FOR T%=&8000 TO &87FF
200   VDU31,17,VPOS-1:PRINT~T%
210   PROCread(T%)
220   IF A% <> T% MOD 256 PRINT"RAM
      error at address ";~T%:END
230   NEXT
240   PRINT"RAM write 2"
250   FOR T%=&8000 TO &87FF
260   VDU31,17,VPOS-1:PRINT~T%
270   PROCwrite(T%,&55)
280   NEXT
290   PRINT"RAM read check 2"
300   FOR T%=&8000 TO &87FF
310   VDU31,17,VPOS-1:PRINT~T%
320   PROCread(T%)
330   IF A% <> &55 PRINT"RAM error
      at address ";~T%:END
340   NEXT
350   PRINT"RAM write 3"
360   FOR T%=&8000 TO &87FF
370   VDU31,17,VPOS-1:PRINT~T%
380   PROCwrite(T%,&AA)
390   NEXT
400   PRINT"RAM read check 3"
410   FOR T%=&8000 TO &87FF
420   VDU31,17,VPOS-1:PRINT~T%
430   PROCread(T%)
440   IF A% <> &AA PRINT"RAM error
      at address ";~T%:END
450   NEXT
460   PRINT"Check complete."
470   END
480
490   DEF PROCmach
500   code%=&A00
510   P%=&A04
520
       [                  OPT0
530                       LDA &70
540                       PHA
550                       LDA &71
560                       PHA
570                       LDA code%
580                       STA &70
590                       LDA code%+1
600                       STA &71
610                       LDY #0
620                       LDA #12
630                       STA &FE30
640                       LDA code%+2
650                       STA (&70),Y
660                       PLA
670                       STA &71
680                       PLA
690                       STA &70
700                       LDA #252
710                       LDX #0
720                       LDY #&FF
730                       JSR &FFF4
740                       STX &FE30
750                       RTS
760    ]
770   ENDPROC
780
790   DEF PROCread(address%)
800   ?&F6=address% MOD 256
810   ?&F7=address% DIV 256
820   Y%=12
830   A%=USR&FFB9 AND &FF
840   ENDPROC
850
860   DEF PROCwrite(address%,byte%)
870   !code%=address%:?(code%+2)
      =byte%
880   CALL(code%+4)
890   ENDPROC
```

LISTING 3

```
10    REM Program 3
20    REM Sideways RAM signature
      code
30    REM to implement star
      commands
40    REM Stephen A. Todd. (c) 1983
50    REM *T will run code from
      &8070
60    REM and *HELP is implemented
70
80    MODE6:VDU23;8202;0;0;0;
90    PROCmach
100
110   DIM buffer &7FF
120
130   !buffer=&4C000000
140   !(buffer+4)=&F828014
150   !(buffer+8)=&73655401
160   !(buffer+12)=&726574
170   !(buffer+16)=&294328
180
190   FOR T%= buffer+20 TO buffer
      +&7FF
200   ?T%=0
210   NEXT
220
230   FOR T%=0 TO 3 STEP 3
240   P%=buffer+&14
250
       [                  OPT T%
260                       PHP
270                       CMP #9
280                       BEQ help
```

write(X,byte). PROCmach in **Listing 1** must be run once before using PROCwrite. ie PROCwrite(&8000,255) will write 255 into the first location in the RAM and PROCread(&8000) will return the same number in integer variable A%.

We can extend PROCread and PROCwrite to a RAM testing program to ensure that everything is functioning correctly. This is shown in **Listing 2.** Type this in and run it. The program will report any errors that it finds.

As mentioned before paged ROM software can be placed in the extra RAM and run with a * command. The signature for a paged ROM must first be placed in the RAM before this can be done. **Listing 3** does this for you. Running the program, pressing BREAK once then typing *HELP (or *H.) will show an entry for Tester showing that the RAM is working correctly. The machine code that is to be run in the new RAM must now be placed in situ. This is done in the following manner:

(i) assemble the machine code into some convenient USER RAM area noting its length. This must be done using offset assembly with 0% and P% (P% set to &8070). This is explained in the documentation supplied with BASIC II.

(ii) use a BASIC FOR loop and PROCwrite to write the assembled machine code into the RAM starting at address &8070.

The * command can now be called by typing *T. An example of this is shown if you run **Listing 4** after running **Listing 3.** Most of the utilities provided in the computer magazines can be placed in this RAM and be automatically protected from spurious memory writes.

Listing 3 can be extended to cope with more than one star command and distinguish between them. Thus more than one utility can be present at one time called by different star commands. The reader is left to experiment with this. PROCread can be used to read out the RAM into memory after the readers new star commands are added which can then be saved on tape or disc. The RAM can be rewritten using PROCwrite at the beginning of a programming session so that your favourite and most useful utilities are always present and are accessed by a simple star command.■

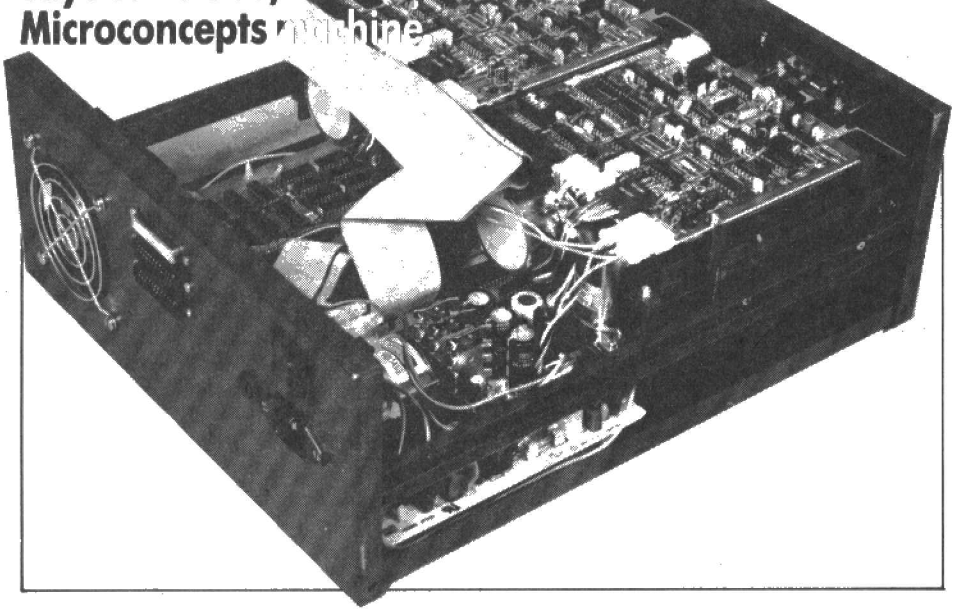
```
290              CMP #4
300              BEQ command
310              PLP
320              RTS
330
     .command
340              PHA
350              TYA
360              PHA
370              TXA
380              PHA
390              LDA (&F2),Y
400              AND #&DF
410              CMP #ASC"T"
420              BNE not_ours
430              INY
440              LDA (&F2),Y
450              CMP #13
460              BEQ ours
470              CMP #32
480              BNE not_ours
490
     .ours
500              JSR &8070
510              PLA
520              TAX
530              PLA
540              TAY
550              PLA
560              LDA #0
570              PLP
580              RTS
590
     .not_ours
600              PLA
610              TAX
620              PLA
630              TAY
640              PLA
650              PLP
660              RTS
670
     .help
680              PHA
690              TYA
700              PHA
710              TXA
720              PHA
730              JSR &FFE7
740              LDX #0
750
     .rep
760              LDA &8009,X
770              JSR &FFE3
780              INX
790              CPX #6
800              BNE rep
810              JSR &FFE7
820              CLC
830              BCC not_ours
840  ]
850  NEXT
860
870  ?(buffer+&70)=&60
880
890  CLS
900  PRINT"Writing code into RAM"''"Address  "
910  FOR T%=buffer TO buffer+&7FF
920  VDU31,17,VPOS-1:PRINT~&8000+T%-buffer
930  PROCwrite(&8000+T%-buffer,?T%)
940  NEXT
950  PRINT'"Pressing BREAK once"
960  PRINT"then typing *H. should show"
970  PRINT"an entry for Tester."
980
990  END
1000
1010 DEF PROCmach
1020 code%=&A00:REM or any safe place
1030 P%=code%+4
1040
      [          OPT0
1050             LDA &70
1060             PHA
1070             LDA &71
1080             PHA
1090             LDA code%
1100             STA &70
1110             LDA code%+1
1120             STA &71
1130             LDY #0
1140             LDA #12
1150             STA &FE30
1160             LDA code%+2
1170             STA (&70),Y
1180             PLA
1190             STA &71
1200             PLA
1210             STA &70.
1220             LDA #252
1230             LDX #0
1240             LDY #&FF
1250             JSR &FFF4
1260             STX &FE30
1270             RTS
1280 ]
1290 ENDPROC
1300
1310 DEF PROCwrite(address%,byte%)
1320 !code%=address%:?(code%+2)=byte%
1330 CALL(code%+4)
1340 ENDPROC
```

LISTING 4

```
10   REM Program 4
20   REM Stephen A. Todd. (c) 1983
30   REM *T will run the following code
40   REM which exists at &8070 onwards
50   REM note: program three must be run first
60
70   PROCmach
80
90   DIM buffer 255
100
110  P%=buffer
120
      [          OPT 0
130              LDA #ASC"Y"
140              JSR &FFE3
150              LDA #ASC"o"
160              JSR &FFE3
170              LDA #ASC"u"
180              JSR &FFE3
190              LDA #32
200              JSR &FFE3
210              LDA #ASC"t"
220              JSR &FFE3
230              LDA #ASC"y"
240              JSR &FFE3
250              LDA #ASC"p"
260              JSR &FFE3
270              LDA #ASC"e"
280              JSR &FFE3
290              LDA #ASC"d"
300              JSR &FFE3
310              LDA #32
320              JSR &FFE3
330              LDA #ASC"*"
340              JSR &FFE3
350              LDA #ASC"T"
360              JSR &FFE3
370              LDA #13
380              JSR &FFE3
390              RTS
400  ]
410
420  MODE6
430  PRINT"Writing code into RAM"'
440  FOR T%=buffer TO buffer+255
450  PROCwrite(&8070+T%-buffer,?T%)
460  NEXT
470  PRINT"Try typing *T now"
480
490  END
500
510  DEF PROCmach
520  code%=&A00:REM or any safe place
530  P%=code%+4
540
      [          OPTO
550              LDA &70
560              PHA
570              LDA &71
580              PHA
590              LDA code%
600              STA &70
610              LDA code%+1
620              STA &71
630              LDY #0
640              LDA #12
650              STA &FE30
660              LDA code%+2
670              STA (&70),Y
680              PLA
690              STA &71
700              PLA
710              STA &70
720              LDA #252
730              LDX #0
740              LDY #&FF
750              JSR &FFF4
760              STX &FE30
770              RTS
780  ]
790  ENDPROC
800
810  DEF PROCwrite(address%,byte%)
820  !code%=address%:?(code%+2)=byte%
830  CALL(code%+4)
840  ENDPROC
```

Concept 09

Users of Z80 and 6502 based machines are stuck with operating systems such as CP/M, but FLEX is best, says S. M. Gee, in this review of the 6809 based Microconcepts machine



Regular readers cannot fail to have noticed that *E&CM's* very own computer project makes a feature of being 6809-based and capable of running the FLEX operating system. These are indeed two very good reasons for being interested in any computer! However, if you don't fancy putting together your own system then you will find this review of the Concept 09 machine, a very high quality factory-built 6809/FLEX system, particularly relevant.

While the majority of the computing community get by using microprocessors of the Z80 and 6502 type and are forced to use operating systems such as CP/M, a privileged (wise?) minority have been using the 6809 running FLEX. There is little point in starting an argument over the merits of one microprocessor or operating system over another — the best is always the one that you know most about. However, this said I would claim that the 6809/FLEX combination has so many advantages that it is difficult to see why anyone is prepared to accept second best!

Seriously though, the 6809 is a powerful 8/16-bit microprocessor that is easy to program in assembler and provides a useful compromise between the sophistication of full 16-bit micros and the economy of the eight-bit family. The FLEX operating system is well-designed, logical and user-friendly. Neither FLEX nor the 6809 are newcomers to the microcomputer scene — indeed FLEX was available on the 6800 at about the same time that CP/M — so you may be wondering why 6809/FLEX users are in the minority? The answer is mainly a historical accident. The first widely available microprocessor was the Intel 8080 and the first operating system for it was CP/M, and being first counts as nearly everything when it comes to micros. Another reason for the lack of 6809 users is the shortage of really good, economically priced systems. Again the reason for this is to be found back in the early days of computing. One of the first micro systems of any sort was the S50 bus 6800 made by SWTPC. At the time the S50 bus was a serious rival to the now well known S100 bus standard and there were and still are a very wide range of plug in cards for the system builder. However, along with its advantages the S50 bus carries some serious disadvantages. Firstly the early S50 systems produced by SWTPC were very cheap but the quality was hardly professional. Other manufacturers, notably Gimix in the USA, produced very high quality hardware but with short production runs and, perhaps in something of an overreaction to the poor quality of the SWTPC systems, the cost was very high. Thus in the minds of many users the 6809 tends to be equated with either very low-cost, low-quality machines or extremely high quality very high cost machines! It seems that what is missing from the 6809 market is a reasonably priced professional quality system and this is where the Concept 09 comes in.

System specification

The Concept 09 is a single board computer, that is it is not based on the S50 bus, which tries to supply everything you need in one box. The only extra that you have to supply to get started is a VDU — but if you don't want to use a separate VDU there is a model available with built-in keyboard and display. The standard unit provides 64k of 4116 dynamic RAM but, as with all 6809 systems, some memory space has to be given over to I/O ports and so 56K is the maximum usable RAM.

A 2K monitor in EPROM provides many of the services that the FLEX operating system needs, along with a simple memory examine and modify command. The monitor makes no attempt to be a machine code debugging facility because once you have FLEX you have access to one of the best machine code debuggers available, DEBUG (see later). As well as the 6809 CPU, the single PCB contains three 6821 PIA's, one 6840 timer and a 1791 disc controller. Two of the PIA's are used to provide four eight bit plus 2 handshake parallel ports. One of these ports can be used to drive a centronics type printer interface; the others can be used for general interfacing, experimentation etc. The third PIA is used to provide additional disc control lines such as drive/density/side select. The disc controller itself will handle single and double density five and eight inch discs. The standard unit comes with a pair of half-height five inch drives with 80 track and double sided drives optional. Using standard 40 track single sided discs gives 175K of storage per drive. Double-sided and 80 track drives increase this to 700K per drive so there is plenty of storage without having to go to eight inch drives. Also the system can handle a maximum of four drives although this is a rare configuration. Of the three 16-bit timers in the 6840, two are dedicated to system use and one is free for user applications. The 6850 provides the serial port to drive the VDU. The baud rate is software selectable from 50 to 19200. An additional RS232 port can be added as an optional extra.

Design features

The hardware specification of the Concept 09 makes it a fairly full function 6809 system. Most S50 bus machines costing twice the price of the Concept 09 have very similar specifications. The overall quality of construction is very high with the main PCB and a switch mode power supply mounted on the base of a very solid steel case. The twin half-height disc drives are mounted across the top of the case, so obscuring half of the main PCB. Ventilation is taken care of with a miniature fan mounted to the rear. The parallel and serial ports are also brought out at the rear of the case. Overall the Concept 09 is a simple straightforward design with no unnecessary frills.

Compatibility

The Concept 09 isn't an S50 bus machine but its software compatibility with such

Concept 09 at a glance

Processor	6809
Memory	64K dynamic RAM 2K monitor.
I/O	6821 PIA (3), 6850.
Disk	40 track, single sided (2) 175K storage per drive.

machines will be an important question for anyone looking for an economical replacement for existing and ageing S50 bus equipment. Software compatibility is mainly determined by the layout of the memory map. The Concept 09's I/O block starts at $E000 which is standard for FLEX systems. However, instead of being divided 'ports' of 4, 8 or 16 addresses, each I/O device occupies memory from $Ex00. So, for example, the first PIA is at $E000, the second is $E100, the 6840 timer is at $E200, and so on. This means that any program that tries to do direct I/O bypassing the FLEX operating system will have to be modified before it works on the Concept 09 — but in practice this should be a very small problem as all the best programs use FLEX anyway! The monitor supplied occupies a different position in memory to either of the two most popular monitors ie SBUG and GIMIXBUG and so there is no compatibility at this level. However, the Concept 09's monitor entry points are fully documented and they are as useful as anything in SBUG or GIMIXBUG.

If you look at the 09's memory map then you will see that as far as FLEX is concerned the Concept 09 is a very standard machine. I tried running programs that were developed on my own SWTPC sytstem some time ago and they worked first time. Although the five inch discs supplied with the system can be used with 80 track double-density the systems formatting program will generate a range of different formats including 35 track, single-sided, single-density discs which are ideal for data transfer to other FLEX systems. There must be very few FLEX systems that cannot read this disc format on a five inch disc and this should be compared to the current confusion of formats on other systems. Unless you have a program that makes direct reference to an I/O port or uses some facility of a specific machine monitor then you should find that your old S50 programs work without modification on the Concept 09.

Software

The Concept 09 will run any software that is available under the FLEX operating system so this section applies to all 6809/FLEX systems! One of the main worries of anyone considering leaving the CP/M world is whether or not FLEX users have access to the same amount of software. The obvious answer is that CP/M has so many users compared to FLEX that it wins on the number of specialised software packages that are available. However, FLEX certainly doesn't lack any of the standard systems and applications packages and FLEX software is usually of a higher quality than CP/M. The reason for the higher quality software under FLEX is not at all obvious but it may have something to do with the use of the very logical 6809 assembly language and the fact that FLEX itself is easy to use. FLEX is a small single user, single tasking operating system taking only 8K of memory to provide sequential and random access files. All of the system commands are disc resident. This means that they only use RAM when they are needed and can be customised without having to delve into the inner workings of the operating system. A very useful feature of FLEX is the printer spooling system. FLEX printer spooling allows text files to be entered into a 'print queue'. Each file in the print queue is sent to the printer while the rest of the system can be used as normal. In this sense FLEX is a limited multi-tasking operating system in that it will handle one user task and one printing task at the same time!

It is impossible to give a complete list of FLEX-based software (you can get a good idea of the more specialised software range by looking at an issue of '68' Micro Journal, an American magazine dedicated to the 68xxx range of micros) but it is worth mentioning a few of the more important items of software. If you are planning to use the Concept 09 as a software development system then there is an excellent range of software. The standard TSC macro assembler serves for most assembly language development and by using macro libraries it can be used as a cross assembler for the 6800, 6801, 6805, 6502, Z80, 8080 and 8085. If you need to develop larger assembly language programs then you can use the TSC relocating assembler and linking loader. A cross assembler producing 68000 machine code is also available. There are a large number of debugging aids including a wide range of disassemblers and simulators. The most useful debugging aid is TSC's DEBUG package. This allows 6809 machine code to be run in a completely controlled environment very similar to that provided by an interpreter. Indeed the best way to describe DEBUG is to say that it is an interpreter for 6809 machine code. As far as high level languages are concerned there are no shortages of good software. TSC BASIC and Extended BASIC are similar to Microsoft BASIC and very fast. There are various implementations of Pascal, C and a Fortran 77 compiler and Forth. For the prospective business or office user applications packages include a number of excellent text processors, spelling, mailing list, data base and accounting packages. This short list of software should at least convince you that FLEX and hence the Concept 09 can be used in any of the applications areas that you might find CP/M.

Hardware expansion

The Concept 09 isn't really designed to be expandable in the same way that a traditional S50 bus system or *E&CM's* own 6809 machine are. However, most of the internal bus signals are available by way of the spare EPROM socket on the main logic board. A range of plug-in peripheral cards can be added to the machine using this method. The biggest restriction on the current design is the lack of any memory management. Many 6809 CPU cards use various forms of memory management to extend their addressing range and allow multi-tasking multi-user operating systems such as UniFLEX to be used. It is possible to imagine ways in which such memory management could be added to the Concept 09 by using its spare PIAs but for the time being the single user Concept 09 is all that is available. (The multi-tasking operating system OS-9 is promised in the near future as an alternative to FLEX).

Conclusion

If you need a good quality and economically priced 6809 system then the Concept 09 is worth looking at. It has all the features required of a full-function but standard FLEX system. My only criticism, and it is a very minor one, is the lack of any real provision for hardware expansion within its box. The Concept 09 is a pleasure to use and I have no doubt that it will help enlarge the size of the 6809/FLEX user community.

The Concept 09 is available from Micro Concepts, 8 Skillicorne Mews, Queens Road, Cheltenham, GL50 2NJ (Tel: 0242-510525). A number of models are available offering different disc formats. The one reviewed (with dual 40 track single sided disc drives) costs £995 plus VAT. The same model complete with VDU costs £1850 plus VAT. ■

SOFTWARE REVIEWS

Adam Denning reviews a selection of the latest utility software.

Disc Doctor

(BBC Models A and B)
Computer Concepts

This ROM plugs into one of the sideways ROM sockets on the BBC Micro and offers a number of extremely useful features to the disc owner. Before installing the ROM, it is important to note that Disc Doctor is only guaranteed to work with the Acorn DFS ROM, and it is reported not to work with any version of the Watford DFS prior to version 1.2, or with any version at all of the Amcom DFS.

Typing *HELP DISC with the ROM installed will cause the ROM to print out a DFS-style list of all the commands available to it and the required syntax for each. Most commands have a number of parameters, some optional, and all are invoked by a * command, followed by the utility name and the requisite parameters.

The commands offered and their effects are as follows:

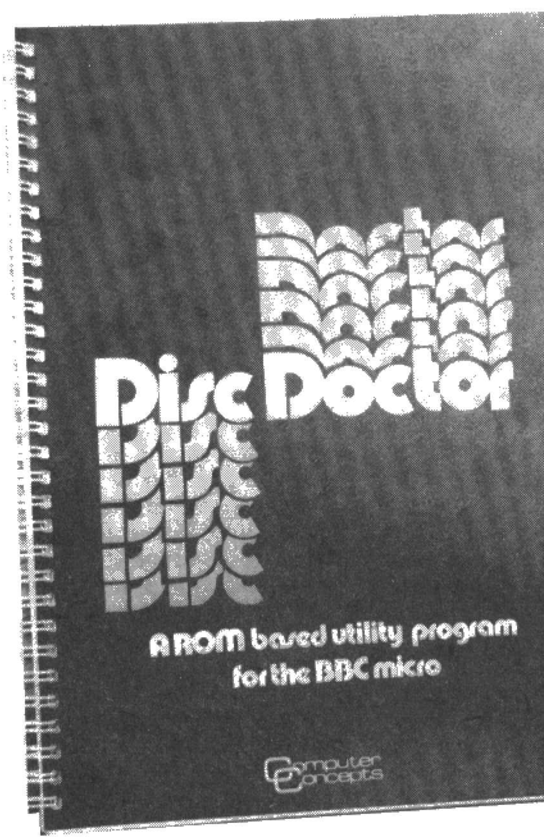
***DIS** which invokes a symbolic disassembler, and allows the user to examine the contents of memory in 6502 mnemonics. Also displayed are the addresses concerned, the machine code in Hex and, if relevant, the ASCII representation of each byte. You may state the start and end addresses of the disassembly, plus an offset which makes the disassembled code appear as if it were located at this offset address. In teletext mode, all this output is arranged in neat columns with different colours for each field. Pressing certain keys allows various options to be taken; 'space' causing the disassembler to follow jumps, B going back in the disassembly by one byte, C only following jumps that do not lead into the language or operating system ROMs, D causing the output to be expressed as data bytes rather than 6502 mnemonics and M reversing this option. Also available are F which causes the disassembly to be continuous, and S which prints out one instruction per keypress.

***DISCTAPE** allows programs to be transferred from disc to tape without all the tedium normally involved in this process. It proves very useful if all the files to be transferred have different names, but could not cope with our intended transfer of Acornsoft FORTH to disc, as all the files on the cassette are called FORTH. Thank you Acorn!!

***DOWNLOAD** loads a named file and moves it to a specified address. This is very useful for relatively large programs that will run out of room if Basic's PAGE system variable is not reset from 1900 to E00 before running.

***DSEARCH** will search any specified or defaulted part of a disc for an ASCII string, in a particularly fast way. If the string is found then the DZAP disc editor (description following) is invoked. The drive number, track and sector numbers are displayed at all times. It was interesting to find that some discs that were formatted using the Torch Z80 disc pack gave 'wrong disc size' errors under this command. This is because the formatting program in the Torch pack is not quite right.

***DZAP** is the interactive disc editor, and allows the user to selectively alter parts of a disc. The contents of a particular sector are displayed on the screen and the user may move the cursor through these bytes, changing the contents as he goes. The new contents can be entered in a number of forms, including ASCII, hex and decimal, and if a change has been made then an attempt to move off that altered sector will cause the ROM to ask if the user wants to save the alterations back to disc. Pressing 'Y' to this prompt will do so, any other key leaving the disc as it was. This is very useful, as it means that one can become familiar with the system before doing anything destructive.



***EDIT** is one of the less useful functions, and simply allows the user to alter the contents of a specified red function key.

***FIND** will list all the line numbers in which a specified string is found. This string can include Basic keywords, but these must be entered in the internal BBC tokenised form, as this is how they are actually stored in the machine. It can only be used on Basic programs.

***FORM** is a very useful formatting program which when held in ROM rather than having to be loaded from disc means that its use will not corrupt any programs already resident in memory. It is comforting to know that this program actually works. After formatting the specified disc with the specified number of tracks, the program then asks the user if he wants to verify the disc. It is safest to reply 'yes', and it only takes a few seconds anyway. This command can also be used to format the discs in a special format that gets over the Acorn DFS restriction of only having 31 files per disc.

***JOIN** as its name implies, joins a number of specified files together, purely by appending one to the other, and storing the resultant file under a specified filename.

***MENU** displays a list of all the files on the specified disc in the directories + and −. As very few people use such named directories until they have installed Disc Doctor, initial use of this command produces disappointing results in that not a lot is displayed! However, once the conventions have been adopted the command can be quite useful because by simply moving the cursor to a particular filename it will load and run that file.

***MOVE** moves a Basic program from one location in memory to another, and is useful for instance for having a number of programs in memory at the same time, each at a different value of PAGE.

***MSEARCH** is very similar to DSEARCH except that it operates on the BBC's memory rather than a disc. One member of staff was particularly impressed with the speed with which one could move through memory with the thus invoked editor, MZAP.

***MZAP** as with MSEARCH is very similar to its disc counterpart, DZAP. It offers the same extensive facilities and it is particularly interesting to move the cursor into zero page or one of the input output memory mapped areas, as one can see the computer in action. Each dynamic byte rapidly fluctuates between all its values – very hypnotic!

***PARTLOAD** allows only a specified part of a disc file to be loaded into any specified area of memory, and although having very little use in Basic programming, it does have applications in such things as screen editors, word processors and the much loved BCPL language.

***RECOVER** is used mainly for recovering sectors directly from the surface of a disc. Any number of sectors starting at a specified track can be loaded into memory and then operated on with MZAP. This command is useful if you have just deleted a file by accident, and you happen to remember where on the disc it started. If you don't then DSEARCH and DZAP may prove useful in locating it for you.

***RESTORE** puts specified sectors stored in the computer's memory back onto the disc from whence they came.

***SHIFT** moves a block of memory from one location to another, and only needs to exist because the designers of the 6502 used in the BBC neglected to put block commands such as the Z80's LDIR in its instruction set.

***SWAP** is used exclusively by the previously mentioned special formatting arrangement, in which two separate directories are maintained on the disc so that more than 31 files can be saved onto it.

***TAPEDISC** does the opposite of the earlier DISCTAPE command, and simply moves a program or programs from cassette to disc without the hard work.

***VERIFY** is automatically invoked by the FORM command, and simply checks that all sectors on a disc can be read properly.

That is the entire list of commands offered by Disc Doctor, and it should prove that this ROM is going to be extremely useful to anyone who has a disc system fitted to their BBC, particularly if they also have the Acorn DFS. **AD**

Toolkit

(BBC Models A and B)
Beebugsoft £27.00

This EPROM from Beebugsoft is yet another sideways ROM for the BBC machine, and is intended to help Basic programmers develop and debug their programs. It comes from a BBC User Group called Beebug, so one should hope that the facilities are not only useful, but they also work.

The toolkit costs £27.00, and comes in a large library case with a small 32 page manual. A whole host of commands are offered, and the manual describes each one in reasonable detail, although it does seem to assume that the user is very familiar with both BBC Basic and the Operating System.

A brief resume of the commands:

*CHECK: used to verify programs and files in memory with those saved on disc or cassette.

*CLEAR: this command will clear all the variables in a Basic program in a similar way to Basic's CLEAR command, but it also clears the integer variables, A% to Z%, setting them to zero.

*EDIT: potentially the most useful command of the lot, this invokes a screen editor in Mode 7. This editor only works on a Basic program, which must be in memory before typing *EDIT, and offers cursor movements all over the place, and a few insertion and deletion commands. Calling this a screen editor is a bit of a cheek, as real screen editors do much more. *EDIT is a cross between the BBC's own editor and that of Wordwise, and unfortunately only marginally more useful than either.

*FREE: this command causes the amount of memory left to Basic to be displayed, as well as the current program size and the values of PAGE, LOMEN, HIMEN and TOP.

*MEMORY: this command is entered with up to two parameters, the start and end addresses, and displays the memory between these points in both hexadecimal and ASCII. Another way of doing this for those who have Acorn DFSs is to * SAVE the block of memory concerned and then *DUMP the file thus saved.

*MERGE: this simply merges a program on disc or cassette with one in memory, without having to go through the tedious process described in the BBC User Guide.

*MOVE: a useful command for disc owners; it enables the user to relocate a Basic program in memory from one PAGE location to another. As the DFS resets PAGE from its original E00 to 1900, some programs may be unable to run on disc-based systems, so by typing *TAPE followed by *MOVE E00, the program is moved to where it would be loaded on a cassette-based micro.

*NEW: this is identical to Basic's NEW command, except that it can be used in programs. Not very useful.

*OLD: as for *NEW.

*ON: this command causes (what Beebug describe as) an 'enhanced error reporting system' to be invoked. In practice, this means that standard error messages will be printed in colour (!), and more importantly, the screen editor mentioned earlier will be entered at the current line. As the current line is where the error occured, this is rather a useful facility and makes debugging a lot less arduous.

*OFF: cancels the above command.

*PACK: this command compacts a Basic program by removing all redundant spaces, and if followed by certain option letters, it will also remove all REM statements and comments.

*RECOVER: attempts to overcome the untrappable 'Bad Program' error that so infuriates even the most seasoned of Basic programmers. In general, it seems to work well, but of course not every program is recoverable.

*RENUMBER: this is just like Basic's RENUMBER command, except that it also gives you the option of renumbering only certain sections of a program. It is hard to see the virtue of this.

*REPORT: this is effectively equivalent to Basic's own REPORT, except that the line number is also printed. It seems fairly superfluous.

*SCREEN: saves a screen in any mode to disc or tape.

*UTIL: this is really a number of commands under one roof so to speak, as there are nine options available. One can search for a string and replace it with another if required, move a number of lines from one location to another within a program, list procedures and functions, print out the values of the integer variables A% to Z%, or list any other variables and their current values.

This toolkit therefore has a large variety of at least reasonably useful commands, all of which work as well as intended. However, one gets the feeling that so much more could be done, and hopefully at a lower price. The EPROM uses the RS423 workspace, so care must be taken if you intend to use both the toolkit and the RS423.

If Basic is still your language and nothing seems to work for you, it may pay you to give this ROM a home. Personallv, I can't spare the sideways ROM space. **AD**

Paged ROM for the BBC micro

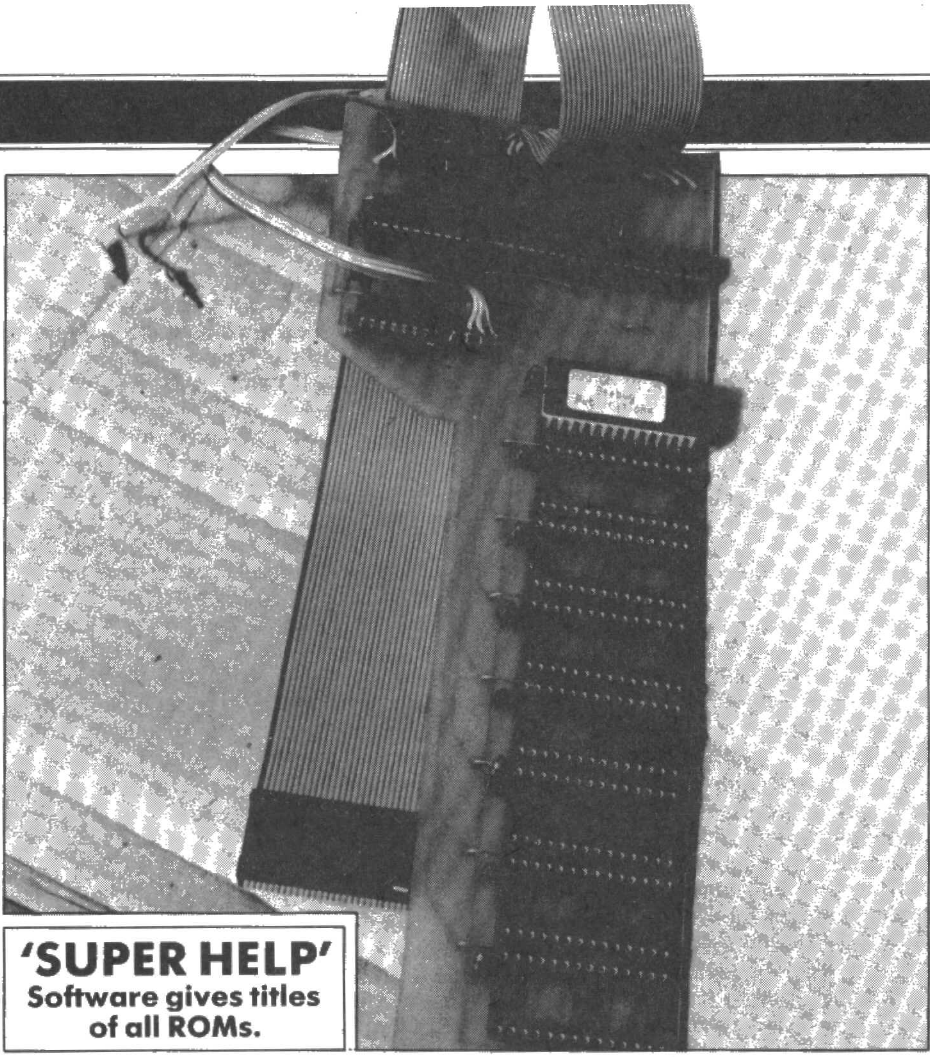
In part two of this project Brian Alderwick and Peter Simpson conclude their description of the hardware associated with the sideways ROM system and describe the operation of their expansion board.

The three to eight decoder ensures that only one paged ROM socket on the extension board can be enabled at one time, but, as on the 74LS139, there is a chip enable input. In fact there are three enable inputs on the 74LS138. One of these needs to be high and two need to be low. In this application the high enable input is connected to Qd and the two low enable inputs are connected to the paged ROM area access signal. Thus the appropriate chip select line will only go low when a memory access is being made to the paged ROM area and the selected paged ROM is on the extension board.

A 74LS245 bidirectional buffer is used to interface between the pages ROMs and the 6502 data bus. This buffer serves two purposes. Firstly, it isolates the 6502 data bus from the load imposed by seven extra paged ROMs. Although the outputs from seven of the eight chips will always be in the high impedance 'tri-state', every ROM will still add about twelve pico-farads of capacitance to each data line, making a maximum addition of about eighty-four pico-farads. This extra capacitance is enough to cause some 6502 microprocessors to malfunction when it is added to the load already present on the data bus in the BBC micro, hence the use of a buffer to hide the extra load from the data bus which only sees one extra load, that due to the buffer itself, and this of course is very small. The second reason for using a buffer, is that it provides a means of isolating the paged ROM output from the data bus when the correct conditions are not present. In this case the line indicating access to the paged ROM memory area is connected to the enable input of the buffer and the most significant bit of the latched value, Qd, is connected to the data direction input. Thus when Qd is high (during access to sockets numbered eight to fifteen) the outputs from the enabled paged ROM are transmitted to the data bus. On the other hand when Qd is low, the buffers

LISTING 1

```
10 REM********************************
20 REM*                              *
30 REM*         SUPER HELP           *
40 REM*                              *
50 REM*   A PROGRAM TO PRINT THE     *
60 REM*   NAMES OF ALL PAGED ROMS    *
70 REM*  PRESENT IN A BBC COMPUTER   *
80 REM*                              *
90 REM*             by               *
100 REM*                             *
110 REM*        Peter Simpson        *
120 REM*                             *
130 REM*            and              *
140 REM*                             *
150 REM*       Brian Alderwick       *
160 REM*                             *
170 REM*          (C) 1983           *
180 REM*                             *
190 REM*******************************
200
210 MODE 7
220 PROCsetup
230 PROCassemble
240 CALL super
250 END
260
270 DEF PROCsetup
280 REM
290 REM INITIALISE VARIABLES
300 REM
310 oswrch=&FFEE
320 osnewl=&FFE7
330 osbyte=&FFF4
340 romselect=&FE30
350 char=&70
360 charl=&70
370 charh=&71
380 romtable=&72
390 romtablel=&72
400 romtableh=&73
410 ENDPROC
420
430 DEF PROCassemble
440 REM
450 REM ASSEMBLE CODE TO PRINT NAMES
460 REM
470 REM RESERVE SPACE FOR CODE
480 REM
490 DIM space 300
500 FOR I%=0 TO 2 STEP 2
510 P%=space
520 [           OPT I%
530 \
540 \ LABEL COLUMNS
550 \
560 .super     JSR title
570 \
580 \ INITIALISE CHARACTER POINTER
590 \
600             LDA #8
610             STA charl
620             LDA #&80
630             STA charh
640 \
650 \ GET ADDRESS OF ROM TYPE TABLE
660 \ TYPE IS ZERO IF NO ROM IS
670 \ PRESENT IN ANY PARTICULAR SOCKET
680 \
690 \ ADDRESS IS RETURNED IN X AND Y
700 \ LOW-BYTE AND HIGH-BYTE
710 \
720             LDA #170
730             LDX #0
740             LDY #&FF
750             JSR osbyte
760             STX romtablel
770             STY romtableh
780 \
790 \ SAVE PRESENT ROM SLOT NUMBER
800 \
810             LDA &F4
820             PHA
830 \
840 \ INITIALISE SOCKET NUMBER COUNTER
850 \
860             LDY #16
870 .s1         DEY
880             BMI s2
890 \
900 \ GET TYPE CODE OF ROM NUMBER Y
910 \
920             LDA (romtable),Y
930 \
940 \ IGNORE IF ZERO
950 \
960             BEQ s1
970             JSR socket
980             STY romselect
990             STY &F4
1000 \
1010 \ PRINT ROM NAME AND VERSION
1020 \ NUMBER
1030 \
1040            JSR print
1050            JSR print
1060            JSR osnewl
1070 \
1080 \ RESET POINTER TO ROM TITLE
1090 \ READY FOR NEXT ROM
1100 \
1110            LDA #&8
1120            STA charl
1130            LDA #&80
1140            STA charh
1150 \
1160 \ TEST WHETHER ALL SOCKETS HAVE
1170 \ BEEN EXAMINED
1180 \
1190            CPY #0
1200            BNE s1
1210 \
1220 \ RESELECT ROM WHICH WAS ACTIVE
1230 \ ON ENTRY TO SUPER HELP
1240 \
1250 .s2        PLA
1260            STA romselect
1270            STA &F4
1280            RTS
1290
1300 \
1310 \ SUBROUTINE TO PRINT TABLE TITLE
1320 \
1330 \ CLEAR SCREEN
1340 \
1350 .title     LDA #12
1360            JSR oswrch
```



'SUPER HELP'
Software gives titles of all ROMs.

act as inputs on the 6502 data bus causing minimal extra loading, but transmitting no information to the bus.

When the paged ROM extension board is connected to the BBC micro, the sockets on the extension board itself will be numbered eight to fifteen and these will corresponmd to IC6 to 13 respectively. The sockets on the BBC's circuit board will then be numbered four to seven with the non-existant sockets zero to three being images of sockets four to seven. Needless to say, the socket on the BBC board into which the extension board is plugged, will not be available to support a paged ROM.

Operation of the paged ROM extension board

Decoding for the sockets on the extension board is similar to that employed on the BBC micro, except that a 74LS138 three to eight decoder is used in place of the two to four decoder on the BBC circuit board. This chip accepts the three least significant bits of the latched paged ROM number, Qa to Qc, and pulls one output low for each of the eight possible input values.

The paged ROM extension board makes use of all four bits stored by the 74LS163 latch on the BBC circuit board and also the decoded address line which indicates when an access is in the paged ROM area.

```
1370           LDX #0
1380 \
1390 \ GET NEXT CHARACTER
1400 \
1410 .t1       LDA string,X
1420 \
1430 \ IF NON-ZERO THEN PRINT ELSE
1440 \ RETURN
1450 \
1460           BEQ t2
1470           JSR oswrch
1480           INX
1490           JMP t1
1500 \
1510 \ PRINT CARRIAGE RETURN AND LINE
1520 \ FEED THEN RETURN
1530 .t2       JSR osnewl
1540           RTS
1550
1560 \
1570 \ SUBROUTINE TO PRINT THE VALUE
1580 \ IN THE Y REGISTER AS A TWO
1590 \ DIGIT DECIMAL NUMBER
1600 \
1610 \
1620 \ PRINT TWO SPACES.
1630 \
1640 .socket   LDA #32
1650           JSR oswrch
1660           JSR oswrch
1670           TYA
1680           PHA
1690 \
1700 \ TEST WHETHER NUMBER IS GREATER
1710 \ THAN 10. IF SO THEN PRINT '1'
1720 \ OTHERWISE PRINT SPACE
1730 \
1740           CMP #10
1750           BPL so1
1760 \
1770 \ PRINT SPACE
1780 \
1790           LDA #32
1800           JSR oswrch
1810           PLA
1820           JMP so2
1830
1840 \
1850 \ PRINT '1'
1860 \
1870 .so1      LDA #49
1880           JSR oswrch
1890           PLA
1900           SEC
1910           SBC #10
1920 \
1930 \ CONVERT ACCUMULATOR TO AN ASCII
1940 \ VALUE AND PRINT IT
1950 \
1960 .so2      CLC
1970           ADC #48
1980           JSR oswrch
1990 \
2000 \ PRINT THREE SPACES
2010 \
2020           LDA #32
2030           JSR oswrch
2040           JSR oswrch
2050           JSR oswrch
2060           RTS
2070
2080
2090 \
2100 \ SUBROUTINE TO PRINT ASCII CODES
2110 \ POINTED TO BY THE CHARACTER
2120 \ POINTER char UNTIL ZERO IS FOUND
2130 \
2140 \ THE Y REGISTER IS PRESERVED
2150 \
2160 \ SAVE Y REGISTER
2170 \
2180 .print    TYA
2190           PHA
2200           LDY #0
2210 \
2220 \ INCREMENT POINTER
2230 \
2240 .p3       INC charl
2250           BNE p1
2260           INC charh
2270 \
2280 \ GET CHARACTER
2290 \
2300 .p1       LDA (char),Y
2310 \
2320 \ EXIT IF CHARACTER IS 0
2330 \
2340           BEQ p2
2350           CMP #&20
2360           BMI p3
2370 \
2380 \ PRINT CHARACTER
2390 \
2400           JSR oswrch
2410           JMP p3
2420
2430 \
2440 \ PRINT SPACE THEN RETRIEVE
2450 \ Y REGISTER AND RETURN
2460 \
2470 .p2       LDA #&20
2480           JSR oswrch
2490           PLA
2500           TAY
2510           RTS
2520 ]
2530
2540 REM
2550 REM SET UP TITLE STRING AT END OF
2560 REM MACHINE CODE.
2570 REM
2580 string=P%
2590 $string="SOCKET       ROM  "+
     STRING$(39,"_")
2600 string?16=10
2610 string?17=13
2620 string?56=0
2630 NEXT
2640 PRINT"START OF PROGRAM IS AT ";
     ~super'"END OF PROGRAM IS AT ";
     ~P%+56'
2650 PRINT"PRESS SPACE BAR TO CONTINUE"
2660 A%=GET:IF A%<>32 THEN 2660
2670 ENDPROC
>
```

MEMORY LOCATION (HEX)	FUNCTION
8000 to 8002	Jump instruction to language entry point.
8003 to 8005	Jump instruction to service entry point.
8006	ROM type byte
8007	Copyright offset pointer
8008	Binary version number
8009 to 80XX	ROM title string terminated by 0
80XX+1 to 80YY	Version string terminated by 0
80YY+1 to 80YY+4	Three characters viz (C)
80YY+5 to 80ZZ	Copyright string terminated by 0
80ZZ+1 to end	User program

The most significant bit of the latched value, Qd, is used to globally enable the sockets on the extension board when it is high and the sockets on the BBC circuit board when it is low. Extra logic, in the form of an OR gate, IC5, is employed to combine this signal with the paged ROM area access signal, to produce a composite signal which is only low when Qd is low and a memory access is being made to the paged ROM area. This signal is sent back to the BBC circuit board to enable the 74LS139 decoder chip, in place of the original enable signal which was the paged ROM area access signal alone.

Acorn have thoughtfully included a command *HELP in operating systems 1.0 and 1.2. This command causes paged ROMs, which are suitably programmed, to issue a message indicating their presence in the machine. Unfortunately, not all paged ROMs respond to this command, Acorn BASIC being amongst them. This makes it difficult to test whether the ROM extension board and associated address decoding are actually working correctly. To facilitate debugging, the program 'SUPER HELP', shown in **Listing 1,** is included in this article. The program has proved to be a useful utility in its own right and gives the socket numbers and titles of all ROMs which the operating system has been able to recognise, regardless of whether or not they respond to the *HELP command.

All paged ROMs conform to a fixed protocol which is shown in the table. Whenever a BREAK occurs the operating system scans each paged ROM socket in turn, ignoring images. When a ROM is found, which is determined by the presence of the copyright symbol '(C)', the ROM type number is entered into the location in the type table corresponding to the ROM socket number. It follows from this that the table is sixteen bytes long, with one byte for each possible socket in the BBC micro. The type bytes indicate what type of ROM is present, whether utility or language etc.

SUPER HELP examines the ROM type table and for every non-zero byte the socket number is printed and the appropriate paged ROM socket is selected so that the ROM's title can also be printed. The title always starts at address &8009. The address of the ROM type table is not fixed between different operating systems, hence the use of the A=170 osbyte call (*FX170) which returns the address of the type table in the particular operating system in use. This allows the program to function properly on operating systems 1.0 and 1.2 as well as any future operating systems.

The program has been liberally commented to explain its operation and will run under both BASIC 1 and BASIC 2. Disc users may care to change line 490 to read '490 space = &900' before running the program. The resultant machine code can then be saved by the command '*SAVE SUPER 900 +100' and run at any time by using the command *SUPER.

Conclusion

The number of paged ROMs for the BBC micro is increasing rapidly at the moment and it will not be long — especially for disc based users — before a paged ROM extension board becomes a necessity rather than a luxury. This project offers the opportunity to construct such a board for less than a third of the price of its commercial equivalent and has been designed to be simple and reliable in operation. ■

LETTERS TO THE EDITOR

Send your letters to The Editor, E&CM, 155 Scriptor Court, Farringdon Road, London EC1R 3AD

Sir,

With reference to the review of the Picturesque "Editor Assembler" and "Spectrum Monitor" in your February 1984 issue, I would appreciate the opportunity to correct some errors of fact in that review.

Our programs were specifically compared with only one other similar package, which is marketed by Hisoft, and which was quoted by your reviewer to have "unbeatable speed" and to be "far far cheaper". According to Hisoft's advertising, their Assembler will assemble 3000 lines of source code per minute. On testing the speed of our Assembler with several listings of over 1000 lines of code, an average speed of 4500 lines per minute was reported. With regard to the price, I accept that Hisoft's product is cheaper, but only by £2.00, which I would not describe as "far far cheaper".

Your reviewer described our Assembler as having "a rather minimal character set". In fact, there are only 8 ASCII characters not available directly from the keyboard, which are those that require a double shift into the Spectrum's 'E' mode, and are some of the least used characters.

Your reviewer deduces that our Assembler fails in its prime function, as "it should cause as little problem aa possible to its user". The operation of our Assembler, in terms of where it stores its source code, and the size and position of its tables and buffers, is completely transparent to the user, and has been designed that way to eliminate user problems.

May I also point out that from March 1984, both our programs will be supplied with Microdrive commands, and will be available to existing owners on a low cost exchange basis (£1.50 per program). The new Assembler will support all Microdrive, Network and RS232 commands and will allow assembly from Microdrive or cassette.

The new monitor will also allow single stepping through ROM or RAM with a full display of Registers, mnemonics, stack contents and definable memory locations, and will also allow Breakpoints to be set in ROM routines.

Both new programs can be customised to work with a number of Centronics printer interfaces.

Martin J. Ridout
Picturesque.

EDITOR'S NOTE
Thank you for your comments on last month's review – we're always pleased to publish both points of view.

Sir,

When I corrected the tape back up with the tips that T. M. Lewis kindly wrote in, I realised that you could still not do more than 36500 in length because he has not seeded register IX with the start address of the header when saving. The following line must be put in as well: 950 Data 221,33,34,111,17,17,0, 175,205,194.

John Fowler
Newton-with-Scales, Preston.

Sir,

Thank your for the helpful projects recently featured for the BBC micro – in particular the EPROM blower and the sideways RAM board. There is a project I would very much like to see prepared, which would incidentally help the owners of other micros as well, namely a parallel printer buffer.

My own particular application is for a BBC micro with WORDWISE driving a Silver-Reed EX44 typewriter/printer via a Silver-Reed I/F44 interface. However, a suitable buffer should serve an Centronics interface. When typing a number of documents it is very frustrating to have to wait for one document to be printed before beginning to type the next. But commercial buffers are too expensive at around £100 for a 16K buffer. Hence the request for a published project.

Thank you in anticipation.

S. E. Wilson
Wokingham, Berks.

EDITOR'S NOTE
As you will see a software based print spooler for a centronics interface is published in this issue. Next month's magazine will include exactly what you require: a centronics printer buffer!

BBC RAM board

Sir,

An ingerant characteristic of the 6502 microprocessor may cause some readers to have write problems with our RAM board which was published in the Dec./Jan. issues of *E&CM*. The 6502 read/write signal is still active when the data lines change and with certain speed memory chips this can cause corruption.

Fortunately, the remedy is simple and is to obtain the read/write signal from IC77 pin 8. At this pin the read/write has been NANDed with the processor clock 2, thus preventing incorrect data from being written to the RAM.

Peter Simpson

READERS' EXCHANGE

A free service for readers wishing to buy, sell or exchange computer equipment. Please send your advertisement, clearly printer, together with your name and address to: Readers' Exchange, Electronics & Computing Monthly, 155 Farringdon Road, London EC1R 3AD.

COLLECTOR seeks very old British transistors, either loose or on circuit cards. High prices offered for rare specimens. Old transistor data books and sheets also wanted. Write for details and/or offer. Mr. Andrew Wylie, 18 Rue de Lausanne, 1201 Geneva, Switzerland.

SURPLUS computer and audio components including complete pcb's, ic's, transistors. The lot for £5 inclusive of post etc. Also isolating transformer (500VA) £6 inclusive. R. Reynolds, 32 Common Approach, Benfleet, Essex SS7 3LA.

SOFTWARE to swap, all originals eg Jetpac, Psst, Espionage Island, CP Forth, Etx, Games Designer for Light Pen, Pool, Chuckie Egg, Bugaboo or Sprite routines. Write for details to Wilson Sinclair, "Gramoir", Carslogie Road, Cupar, Fife, Scotland. Phone: Cupar 54301 after 4 p.m.

WANTED - Ohio scientific boards; any of CA20, CA22, CA25, 555, 541, 520. C. Allfrey, Quires Green Cottage, Willingale, Essex CM5 0QP.

WANTED – Centronics printer, 737-1 or 737-2, in good order. Details to D. Loverseed, 38 South Parade, Bramhall, Stockport, SK7 3BJ or phone 061-439-4841.

IBM 3982 "golfball" selectric printer with parallel port interface. Requires software to run from ASCII – program supplied for Z80. Complete with several typeheads, a few spares and IBM manuals. Has mechanical fault that requires attention. (Runs, but letters misaligned). £65 to knowledgeable enthusiast (£50 if collected). Write to JDR, Westowan, Porthtowan, Truro. Do not send money until sale agreed.

BBC Disc Interface kit. Brand new £75. Ferguson video/audio interface kit for TX10 chassis, unused £20. Micronet ROM version 1.40 £15. S. R. Linter, 1a Bull Lane, Boughton, Kent, ME13 9AH. Phone Canterbury 750600.

SONY ICF7600D synthesized receiver, latest model, new and boxed. AM, FM, SSB. Swap for 2M transceiver, preferably multimode. Cash adjustment considered. Also consider 70cm rig. Phone Glasgow (041) 637 0808. Paul Corrigan, 112 Clarkston Road, Cathcart, Glasgow G44 3DH.

ATARI ROMs, Star Raiders, Mountain King, River Rescue, tapes, controller 16K, Nukwar 16K, Shootout at OK Galaxy 16K, ROMs £15, Cassette £10 each or will swap for Atari 850 Or Why Kim 0268 558037 after 6 p.m. K. S. Day, 96 Plumleys, Pitsea, Basildon, Essex.

FOR SALE: Printer Okidata CP110 commerciaL grade matrix printer (Ex BBC) Centronics and RS232 interfaces, wired for Dragon 32 but works most micros, with manuals £90. G6HQK Wolverhampton (0902) 69285. Mr. R. Keatley, 4 Himley Close, Willenhall, West Midlands WV12 4LX.

FOR SALE Pascal-T ROM plus full documentation for BBC Model B. Cost £68, sell £30. Phone Pontypool 4104 after six. 34 Afon Close, New Inn, Pontypool, Gwent.

FOR SALE: TRS-80 Model One Level Two and Interface, 48K RAM, Monitor, Cassette Recorder, Software, books, Hi-Res Graphic Unit. Sell for £450. Also BBC Model B Modem with Micronet software, sell for £50. Also 3 BBC games, swap for any joystick compatible games for BBC B. Phone 061 368 7145. David Boden, 41 Silverton Close, Hattersley, Hyde, Cheshire SK14 3DG.

ETI/POWERTRAN Cortex Computer. Fully useable 64K RAM – expandable to 1 Meg – plus 16K screen RAM. Uses Texas 9995 16 bit MPU. Ideal real time R+D/Control or business use. Bargain £250. Tel: Chorley (02572) 69172. Mr. M. Flanagan, 118 Wymundsley, Astley Village, Chorley, Lancs PR7 1UT.

CENTRONICS ASCII 779 Dot-Matrix Printer, V.G.C. c/w. a second printer for spares and workshop manual £100. Bill Coombes, 33 Clarence Park Road, Boscombe, Bournemouth. Tel: 0202 424092.

FREE

An EMAP Publication

Your ROBOT

BRITAINS FIRST ROBOTICS MAGAZINE MARCH 1984

INTRODUCING TOPO

HOME HELP OR EXPENSIVE TOY?

ROBOTS IN THE U.S.A. - THE LATEST NEWS

IN CONTROL WITH COMMOTION'S BEASTY

HOBBY ROBOTS - A BRIEF HISTORY

RESEARCH - WHO'S DOING WHAT & WHERE

YOUR ROBOT
2nd FLOOR
155 FARRINGDON ROAD LONDON EC1R 3AD

EDITORIAL

Reaction to the first issue of **Your Robot** has been very favourable, indeed in view of the interest shown in the magazine we have decided to publish it monthly. Hence the earlier than schedules second issue of **Your Robot** that you're now reading.

We're also making **Your Robot** available on subscription – page 5 carries the details.

Plans for future issues of **Your Robot** are well advanced and include a series of articles describing the construction of a low cost, mobile robot complete with manipulative arms.

In next month's issue we are to launch an exciting competition with valuable prizes and a grand final in London later this year.

Your Robot is definitely THE magazine for anyone with an interest in the field of robotics.

American activity

A report on the American robot market appears elsewhere in this issue of **Your Robot** and from this it is apparent that there is plenty of robotics hardware available in the States. Many of these devices are becoming available in the UK – Hero has been here for some time while Topo has just been launched. The RB5X, a very impressive little robot, is also to be imported by CGL in the near future.

All this activity tends to confirm the opinion that 1984 will be the year of the robot.

In next month's issue – the low down on control languages and the 'birth of a robot'.

Editor: Gary Evans
Assistant Editor: William Owen
Contributing Editors: Gary Herman, Peter Matthews, Richard Moyle
Telephone: 01 833 0846
Advertising Manager: Clare Fullerton
Executive: Richard Jansz
Classified: Andy King
Telephone: 01 833 0531

Contents

IN CONTROL WITH BEASTY

Beasty is a small, rugged unit, that allows direct computer control of servo motors. Ken Alexander has the details.

Sooner or later, anyone interested in the practical side of robotics will come across the need to exercise computer control over motorised systems. There are many different types of motor available to the robot builder and the choice of unit for a particular application is critical if the assembly is to perform as predicted. Of the various types available, many require specialised drive circuits to provide the correct control voltages – seldom anything as simple as a steady DC voltage. **Your Robot** will, in the near future, publish an article describing the various motor types and indicate their suitability in various applications as well as describing any special drive requirements of specific motors. Of the various motor designs however one of the most important is the servo.

Essential feedback

A servo unit consists essentially of a high quality motor driving a cam by way of a set of gears that also turns an integral potentiometer. It is the potentiometer, or pot, that provides the essential ingredient of many robot systems-feedback. The pot's wipers are connected to a steady DC voltage which causes an analogue voltage corresponding to the pot's (and hence the cam's) position to appear at its wiper terminal.

A grossly simplified view of a servo mechanism's operation would be to assume that the computer controlling the servo first accepts a number corresponding to the desired position of the servo. The computer would then compare this with the physical position of the cam as indicated by the voltage appearing on the pot's wiper. The next step would be to apply a voltage of the correct polarity to the mechanism's motor while continuously monitoring the feedback signal. As the position of the cam neared the desired position, the magnitude of the drive voltage would be reduced, allowing the motor to take up its station without overshoot. The software overhead to achieve even this simplified view of operation would, without the intervention of any hardware interface, be quite high and involve quite a lot of processor activity. It is these problems that the Beasty is designed to overcome.

Enter Beasty

The Beasty itself is a small, under 40 x 60 x 20mm, black box, with a series of connectors along its top edge. The small size of the unit, together with the way in which it exercises control over the servos (of which more in a moment) would seem to indicate that its design owes a lot to the field of Radio Control electronics.

The Beasty is connected to the BBC micro by a three core cable that carries the +5V from the micro's auxiliary power socket and one of the user port's lines to Beasty. The fact that control of Beasty is via a serial data stream means that there should be little to stop the link being implemented by a infra red or radio control channel – a very important consideration in many systems.

Beasty can control up to four servos, each of these being connected to the unit via a three wire link. Commotion supply a range of servos, those supplied with the review being the standard model from the extensive Futaba range.

"... the interactive control software allows up to four servos to be simply and accurately positioned ..."

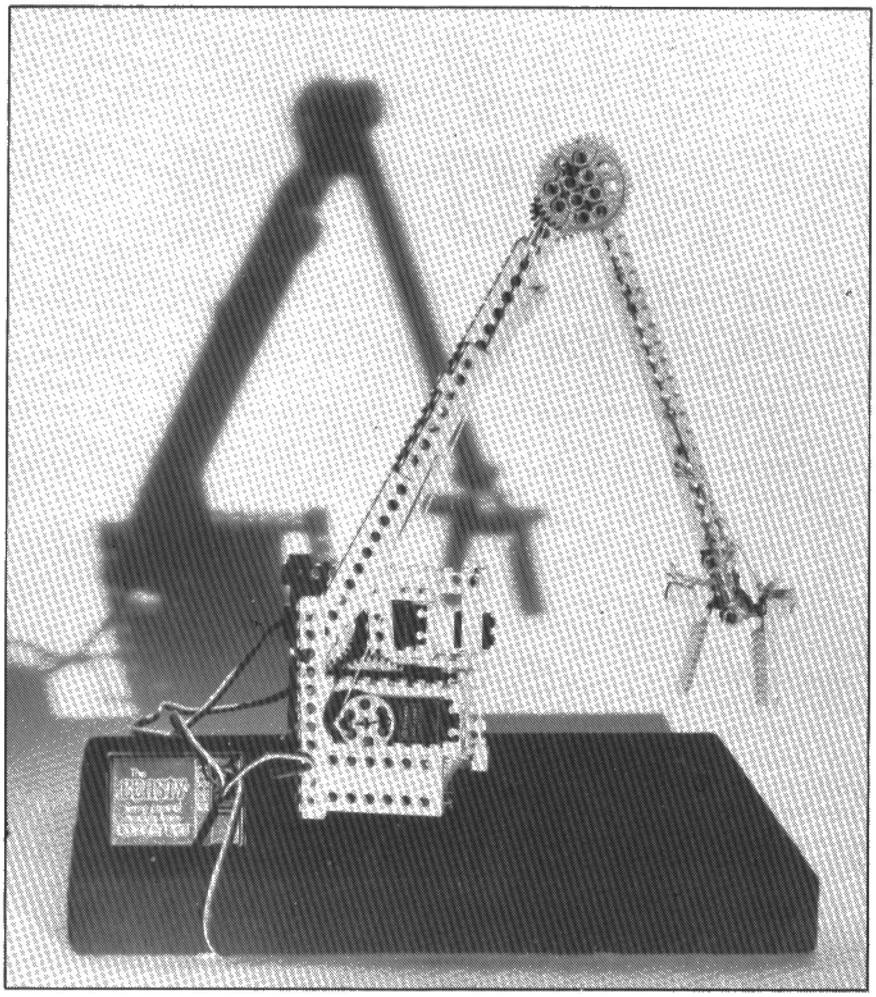
Software suite

The Beasty is supplied with cassette based software suitable for use with OS versions 1.0 or greater. The first program on the tape is ROBOL, described as an Interactive Robot Controller.

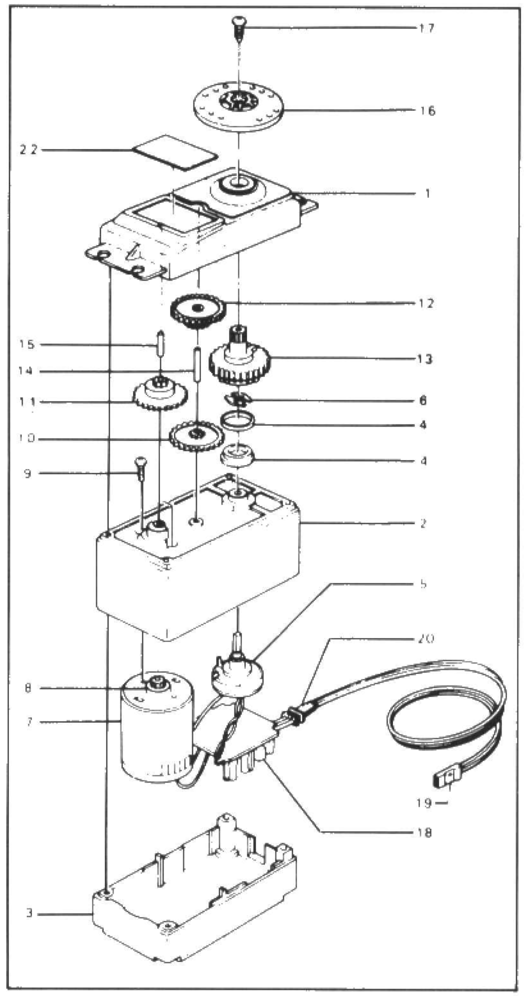
The program allows the position of each of up to four servos to be specified by a number in the range 0 to 996. Upon loading and running the software, the first line of the screen will display the following

```
1    JUMP 500  500  500  500
```

The values shown correspond to the positions of the four servos. Pressing the 1 key will increase the value associated with servo 1, instigating a corresponding movement of the servo, while key Q



Prototype of Commotion's low cost robot arm with Beasty in control.



Exploded view of a typical servo assembly.

will decrease the value of the number. Keys 2 and W, 3 and E, and 4 and R have the same effect on the remaining servos. Pressing SHIFT at the same time as any of these keys will cause the units to change position more rapidly. Having altered the initial values of the servos, pressing RETURN will store them and display the next program line

1	JUMP	0	996	450	600
2	MOVE	—	—	—	—

the keys described above, can now be used to adjust the values of this new line.

1	JUMP	0	996	450	600
2	MOVE	756	—	560	—

A dash would indicate that the corresponding servo is not required to alter its position.

Once a basic movement sequence has been built up it can be saved to the current filing system for later use or modification.

Refinements

The straightforward operation of ROBOL as described above can be augmented in a number of ways. For example, the MOVE command can be replaced by a JUMP statement by pressing f0 at the appropriate line – this will cause the servos to move much more rapidly. Delays and waits can be incorporated in the sequence, either explicitly or by modification of the parameters associated with the JUMP and MOVE commands.

Machine code drivers

In addition to the versatile ROBOL language, machine code users are catered for by a 256 byte, relocatable driver routine.

Used from BASIC this takes the form

```
X%=<servo 0-3>:
Y%=<new position0-255>;
CALL DRIVER
```

another refinement available from machine code is the option to either apply power to the servos continuously, essential if the servo is subject to a continuous turning force, or to power the device only when it is required to change position, thus saving power and avoiding any possibility of hunting.

Lending an arm

Commotion are about to launch a manipulative arm suitable for use with the Beasty and the standard Futaba servos. The price should make the arm an attractive proposition for many budding robot builders.

Last word

Beasty is a well made product that allows computer control of servo assemblies without having to resort to tricky software routines or complex electronics. It should appeal to anyone wishing to experiment with computer motor control. ■

A HISTORY OF HOBBY ROBOTS

Designs for hobby robots have appeared, if somewhat irregularly, since the 1950's. D. Maker looks back at 30 years of activity in this field.

In the development of Hobby Robotics there have been really four historical stages. The first was the introduction in the early 1950's of small powerful permanent magnet electric motors. The second in the early 1960's was the availability of the transistor, next in the early 1970's was the advent of integrated logic circuits and the fourth the introduction in the early 1980's of small powerful microcomputers.

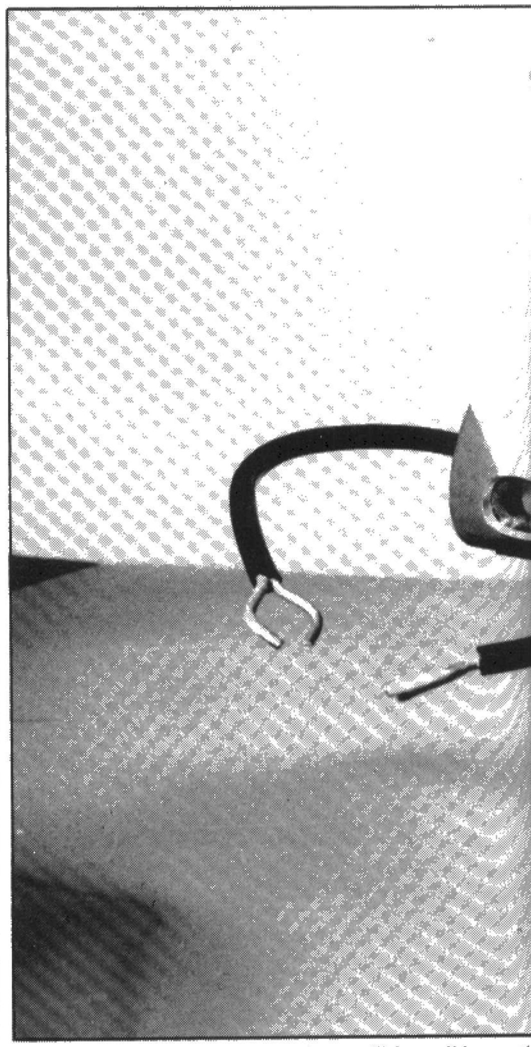
Getting moving

The small electric 'model' motors of the 1950's while by no means as powerful as those available today, nor even as cheap, did allow hobbyists to make small electric powered vehicles. The magazine *Model Maker* in the early 1950's ran a series of articles by Peter Holland on how to construct various 'space models'. One of these 'Mr Robotham' was a two foot high 'walking' robot which had two electric motors driving wheels in its feet enabling it to shuffle along like many toy robots today. Another of this series was 'Teal' which had three legs powered by an electric motor and this really walked. Mr. Holland later constructed a larger Mr Robotham which was radio controlled and had a mechanism in one hand whereby it could sign its name.

These early machines were constructed from the usual modelling materials of the time, that is bolsa-wood, ply-wood, paper clips, brass shim, cardboard and rubber bands which are all easily worked with the limited tools available to the hobbyist. (Unfortunately most people nowadays seem to overlook such materials which with the use of such modern wonders as five minute epoxy and cyanoacrylate super glues allow robot chassis to be made extremely easily).

One other well known early robot was the Meccano walking (shuffling) robot powered by an electric motor working through cranks to move the legs.

The issue of 'Radio Constructor' that introduced a robot called Cyclops back in 1972.


Peter Holland's Teal managed a credible walking acti

Under control

With the introduction of transistors and integrated circuits second and third stage hobby robots could act on their own. The earliest published one to my knowledge was 'FRED' in *Radio Control Models and Electronics* December 1964. Again by Peter Holland, this was a simple light seeking device using two electric motors, two transistors and a light dependant resistor. The LDR was mounted in a tube on the top of the vehicle and rotated as FRED moved forward, simple mechanical switches and the two transistors worked to turn FRED in the direction of any strong light.

In March 1969 *'Practical Electronics'*

started a series on EMMA by G. C. Brown. This design constructed mainly from Meccano had two electric drive motors, port and starboard light sensors and a 'brain' made up from 24 transistors. As well as being able to seek out light 'she' could sense obstructions by monitoring motor loading. When an obstacle was detected, random logic would take over so EMMA could wriggle out of difficulties. A later development which involved doubling the brain size allowed the machine to be trained to accept higher than normal motor loadings and so carry loads.

lt featured three legs driven by a single electric motor.

The next to appear was probably 'Cybernetic Cynthia' by L. C. Galitz in the *Radio Constructor* for June, July 1970 but unfortunately I do not have any information on this machine. 'Xee', published in *Practical Electronics* for June 1971 and following, was a development by G. C. Brown of his EMMA in its first stage but this time using integrated circuits instead of transistors. However, construction was implemented the hard way using an aluminium chassis.

"... the introduction of transistors heralded third stage robots ..."

Another interesting 'animal' was 'Cyclops' by L. V. Galitz published as a series in the *Radio Constructor* from July 1972. Again the main feature of its abilities was phototropism, that is seeking out a source of light. This was sought out by a photosensor in a scanning tube similar to that used in FRED. Obstacles were detected by fender bars activating microswitches and avoidance routines were built into the 31 transistor brain. Cyclops could be trained in a similar fashion to EMMA but this time the association was a little more complex in that it could either learn to associate magnetism with light or to associate magnetism with touch so two entirely different responses to stimuli could be elicited depending on training.

"... the main feature of many designs was phototropism, that is seeking out light ..."

1975 saw two more animals published in the June issue of *Elektor*. The 'Moth' based on a toy lorry turned left unless it could see light to the front in which case it then turned right until the light disappeared, a mercury switch detected bumps which caused the vehicle to turn the opposite way and back up a little. More complex was 'Beetle', this used the same type of light seeking logic but in addition was sensitive to ambient light and would fall asleep for a short time if in the shade, a loud clap would awaken it. Bumps were detected by microswitches and it could learn to associate a clap with a bump and hence take avoiding action at a clap alone.

Also in 1975, and coincidentally for June, *Everyday Electronics* published a design for a white line following vehicle. Unfortunately for Robot addicts that seems to have been that in Britain for the next four years.

Resurgence

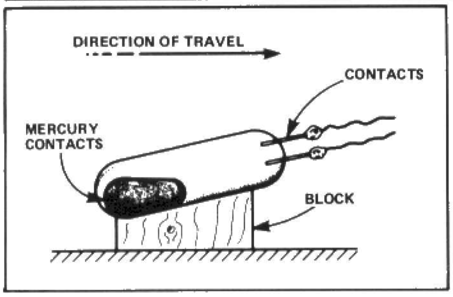
We are now up to 1979 and in November *Hobby Electronics* started a series of articles on 'Hebot' which was also offered as a kit of parts by Remcon. This surely was a turning point for to my knowledge never before had an electronic robot been available off the shelf at least not at hobbyist prices and on this side of the Atlantic. Constructors were not slow to take advantage of this for before the third article in the series had appeared over two hundred chassis had been sold.

Hebot itself consisted of a 10 inch hexagonal aluminium base plate with two wheels on a diameter and fore and aft castors allowing it to turn on the spot. The drive motors and gearboxes were high quality model aircraft servo assemblies without the normally associated electronics and only drew about 150mA each so giving Hebot a running time of between one and two hours from a set of AA nicads.

The electronic logic (obviously by now ICs) worked on a series of priority levels which could be switched round (at least with a soldering iron). The lowest priority level of activity was a random walk followed by two published searching modes, phototropism (again!) and inductive loop tracking. For those of you unfamiliar with inductive loops a long length of wire is laid out in the path the robot ought to take. Both ends are connected to a powerful signal generator and the vehicle tracks the wire using search coils connected to sensitive amplifiers in which detect the signal in the wire loop.

At a still higher priority level was the obstacle avoidance routine activated when objects were detected by feeler wires actuating microswitches.

The four highest priority levels were to be for other senses and external control but if anybody did any work on these nothing seems to have been published.



A mercury switch can be used to initiate a programmed reaction if a chassis runs over a bump.

The final stage three vehicle was Timbug II in *Elektor* for June 1980 (June again, maybe its all those dark winter nights). Timbug II used four 555 timers, four transistors and two relays for control. When it detected obstacles using ultrasonics it backed up and randomly turned right or left before continuing on its journey.

This was probably the first time ultrasonics had been used for obstacle detection on a published design for a hobby robot.

Review

One thing that does stand out regarding these early machines is that, with the exception of those designed by Mr. Holland, from design and construction points of view most had very poor mechanics.

If a robot is going to be any use even as something to play with to learn about robotics it is essential that construction be reasonably sound so that it doesn't keep falling over and/or bits keep falling off. It is not necessary to have a machine tool to achieve this, going back to what I said at the beginning wood is almost an ideal construction medium for hobby robotics, for besides being strong and glueable (preferably epoxy) it provides reasonable bearings for shafts. However this is a digression. (Some would say obsession).

In these days when every model and hobby shop sells small electric motors and electronic components are widely available to the hobbyist it is difficult to imagine the problems that confronted the early robotics hobbyist, especially with regard to electronic components.

In the early to mid 1960's one transistor cost on average about six shillings which when compared with the average cost of a hobby magazine of two shillings was a lot of money. Even in the mid '70's a BC109 transistor still cost half as much as a magazine. Consequently one should not be surprised that these machines are all quite primitive in their capabilities. Light is one of the easiest things to get a machine to look for and light is cheap so it is not unreasonable that this was used to provide a goal for their behaviour. Also the use of light in this way has a historical precedent in Dr. Grey Walters' experiments with electronic animals in the early 1950's.

Besides a goal for their behaviour all of these machines under electronic control, apart from FRED and the white line follower, have some form of sensor for detecting obstacles so that it is possible to initiate a self preservation routine. The self preservation serves a double purpose first of all it protects the robot and secondly, where the routine is simply to withdraw it, protects the environment.

"self preservation serves a dual purpose".

No matter though, the sensors and control patterns, were built into the hardware all these early machines lacked flexibility. Once the vehicle had been built changing the rules for its behaviour meant either getting out the soldering iron or building in a learning feature so that simple associations could be made, for example Cyclops could learn to associate light with magnetism.

With the introduction of small powerful microcomputers however, came the opportunity to build the rules for behaviour into software. Hence the complexity of action of fourth stage hobby robots is enormous, and because of this I shall leave coverage of them until next month.

US ROBOTS

Surprisingly, the level of sophistication of the US market in robots and associated products, is not far in advance of activity here in the UK. Gary Evans visited the Las Vegas Consumer Electronics Show earlier this year and reports on current American trends.

The CES provides a twice yearly opportunity for retailers and manufacturers of a broad range of consumer electronic products to gather together and, in the case of the winter show, for buyers to place orders for the products that will fill their shelves over the summer period. Of all the categories of product on display at the Las Vegas show, comparisons with previous events showed a major growth in the amount of computer related merchandise available. As part of this growth, a number of companies were showing robots aimed at the consumer/educational markets.

The dividing line

Conversations with the various companies showing robots revealed that the market was seen as being divided into two, possibly three, distinct categories.

Hubot – an example of an up-market, consumer robot.

Of these, the major division was between educational and 'up-market' consumer products with the third, more nebulous, division being the promotional/entertainment robot.

Taking each of these markets in turn, the educational robot is designed to provide a useable robot chassis as a reasonable cost. This sort of robot is typified by a product range that includes a base unit (chassis and some basic sensors) and a range of add-ons that can be added to enhance the Robot's capabilities (eg manipulative arms and sophisticated sensors). Manufacturers of this type of robot also tend to pay more attention to the problems of controlling the robot and usually provide an all important software support facility.

The RB Robot Corporation's RB5X is a good example of this sort of educational/experimental robot although it is evident that this particular product overlaps to a considerable extent with the next group – the 'up-market' home robot.

The Hubot is a typical example of this sort of device and here the appeal is much more to the "I must be the first on the street to have a robot" brigade. The marketing of this sort of product emphasises the 'Executive Toy' aspect of the robot with such suggested uses as using it to greet party guests while serving them with a dry Martini. As part of this overall design philosophy, robots in this group often incorporate such consumer luxuries as hi-fi systems and TV sets and have a price tag to match their specification.

The last category of robots is the group with the longest history – promotional robots. These robots, as often or not fashioned in the form of oversize drink cans or beer glasses, have been popular in the States for many years. In many cases, they are not endowed with much in the way of intelligence and are controlled by means of fairly standard radio control systems. Some companies in this area are however finding that their experience in the design and construction of chassis systems is now a valuable commodity. Indeed one such company, the Robot Factory, has now produced a general purpose robot aimed at the educational sector.

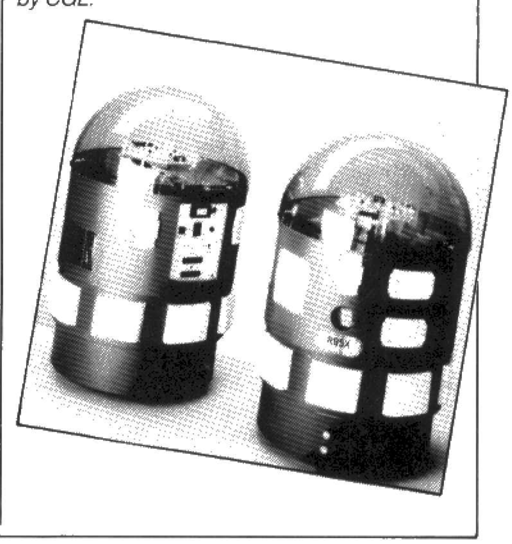
Applications

The current level of sophistication in robot design means that the general purpose machine capable of washing the car on Sunday morning before preparing the roast dinner is still some way off. With the current restrictions bourne in mind however a few useful ideas as to applications are starting to emerge.

The front runner among these is the use of robotics in home security. A robot equipped with a smoke sensor and an infra red detector to establish the presence of fire is quite possible. Adding an ultrasonic range finder for the detection of an intruder is also technically quite straightforward. A mobile robot equipped with such facilities with, possibly, the addition of a cordless telecom system to contact a remote authority is a practical proposition. Indeed, considering the cost of offering the same protection throughout a building using conventional systems, the robot could prove to have the financial edge.

Other applications include robots to take care of vacuuming the house, watering plants and robot nannys to keep the kids amused. Plenty of ideas and a lot of research effort could see many of these applications being accepted as common place during the next couple of years. ■

RB5X – A useful robot soon to be imported by CGL.



Robots–past, pre

In the second instalment of his article, Gary Herman looks at some of the currently available home robot systems.

Beginners might prefer to buy one of the buggies or robot arms currently being offered at reasonable prices by companies catering to the education sector. The simplest buggy available is Colne Robotics 'Zeaker Micro-Turtle', designed for use with the BBC Micro and the Spectrum. This is a no-frills, two motor device using its own power supply, including a retractable pen for drawing Logo graphics and six switch-type touch sensors.

Using the BBC A-to-D input, it would be a simple matter to build an elementary vision sensor for a device like the Micro-Turtle. A photoresistor (for example, the ubiquitous ORP 12) is used as one arm of a potential divider connected to one channel of the A-to-D input. The BASIC function ADVAL then returns a value dependent on the light falling on the photo-resistor. A few program lines are all that's required to direct the Micro-Turtle towards or away from a light source.

The Edinburgh Turtle from Jessop Microelectronics can also be used with the BBC Micro and the Spectrum, while a device known as the BBC Buggy (from Economatics) already includes visual sensors enabling it seek light sources, negotiate obstacles and follow lines.

Robot arms are available to the hobbyist in a variety of models — from a 'meccano'-style crane called The Beasty (supplied by Commotion) to Cyber Robotics' £700 Cyber 310 robot, controlled from any 8-bit parallel port, capable of lifting 250 grams and supplied complete with forth compiler and Cyber's RoboForth and BASIC routines. The arms themselves are all more-or-less standard. There are four standard geometries, but one has come to dominate. Almost all commercial robots use revolute geometry in which a multiply-jointed arm rotates about an axis like a human arm rotating about a human waist. The other geometries are cartesian (planar motion only); cylindrical (planar with yaw); and polar (using a length-adjusting arm with yaw and pitch) **(Figure 1).** For positioning tasks in three dimensions, revolute geometry is close to ideal, enabling the arm to manoeuvre round obstacles and approach objects from a variety of different angles. (Programming six joints, however, is a complicated job).

SMALL ROBOTS

Manufacturer	Origin	Model	Type	D of E	Lifts (g)	Price (£)
Colne Robotics	UK	Armdroid I	J	5	300	400
		Armdroid II	J	5	2000	1500
Systems Control	UK	Smart Arms 4E	J	4	50	480
		Smart Arms 6E	J	6	200	780
Powertran	UK	Micrograsp	J	5		200
		S-101	C	5	1500	740
		P-101	J	6	2000	950
Cyber Robotics	UK	Cyber 3	J	5	250	650
Microbot	USA	Alpha	J	5	680	$10000
		Minimover	J	5	450	$2000
Sandhu Design	USA	Rhino XR-1	J	6		$2400
Mitsubishi	JAP	Movemaster	J	5	500	
Patscentre	UK	Yes-Man	C	2x3		10000
Seiko	JAP	700	C	4	1000	9000
Pendar	UK	Placemate 2	J	5	2000	13000
Microbo	SWZ	MR-01 Souris	P	5	100	expen-
		MR-02 Castor	P	5	1000	sive

Notes:

Types J = jointed, or anthropomorphic geometry
C = cylinder
P = polar

d of F: Number of degrees of freedom of the robot (the Yes-Man has two arms)

Table showing some of the range of small robots now available.

"... the classic prototypes of robots are hand-eye machines and maze solvers ..."

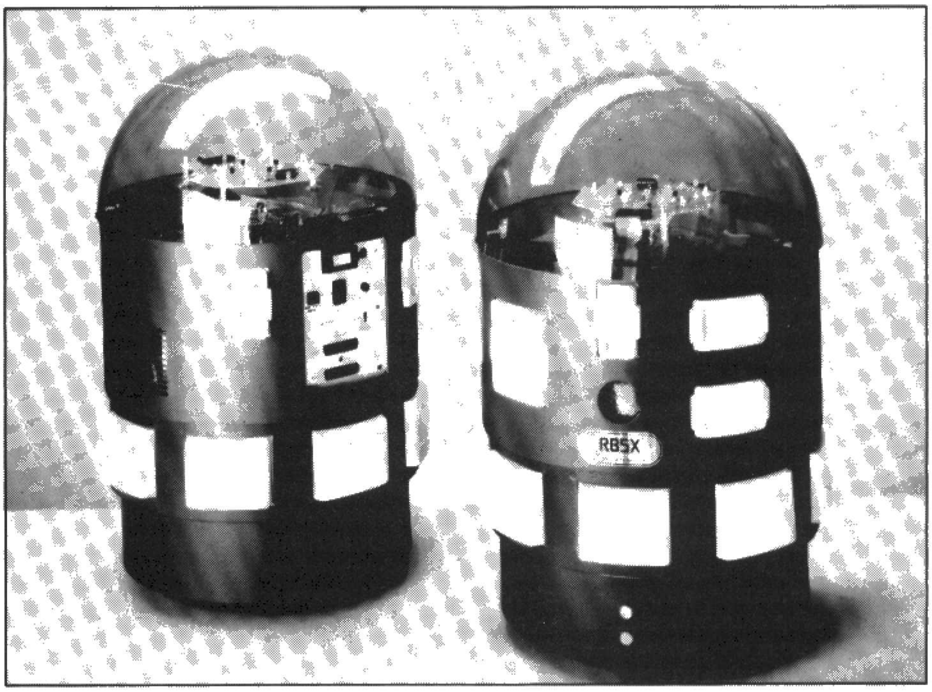
As with buggies, touch and vision sensors can be added to the robot arm — microswitches can be used to detect obstacles, pressure sensors to determine grip and, say, a video camera and image processor used to recognise objects. Similarly a speech-recognition system could be attached to a buggy or an arm to give it directions — several are available, and they are getting cheaper all the time. The Texas Instruments' SP1000 chip, for example, recognises **and** synthesises speech using linear-predictive code and sells at a volume price of only $7.50. (This is the chip used by the speech add-on for the TI99/4A computer).

The problem with all this is software. For most applications, hobbyists will need to write their own software and this task is not facilitated by the predominance of BASIC among microcomputers. Fortunately, robot-specific languages (including a number of enhanced BASICs) are beginning to appear — Microbot's Armbasic, Intelledex's Robot Basic, IBM's AML, Unimation's update of its classic VAL language, Cyber's RoboForth and American Robot's AR-Basic, among them. Robot arms are frequently programmed by use of a teaching pendant — which takes the machine through a series of steps, committing each one to memory — but

esent and future

intelligent robot behaviour requires complex suites of programs to co-ordinate, interpret and learn from a range of inputs and outputs. Software development is undoubtedly the most fruitful area for enterprising hobbyists.

The two classic prototypes of robot devices are hand-eye machines and maze-solvers. Mechanically, the maze-solver is a simple motor-driven wheeled device capable of turning and forward and reverse motion. It is said to have two degrees of freedom — which means, in effect, that given the set of final positions for the maze-solver two independently operating motors could drive the machine to any specified position. A hand-eye required to operate in three dimensions would need at least three degrees of freedom. If the orientation of the hand is taken into account, the machine would need six degrees of freedom — three representing longitudinal, latitudinal and perpendicular motion and three representing rotations about these axes (roll, pitch and yaw, respectively).



Notable work

Sensors attached to maze-solvers and hand-eyes allow them to perform tasks within their environments. Work done in the sixties and early seventies — notably at Stanford University, Edinburgh University and the Massachusetts Institute of Technology — employed such devices in artificial intelligence research. Typically, a hand-eye would be required to manipulate objects in a so-called 'blocks-world': children's building blocks, coloured pyramids, boxes and cylinders. It might also be required to answer questions about its small world, make deductions and learn from experience. If mounted on a maze-solver type device, the whole system would be required to pick paths through blocks and to fetch and carry on command.

A simple articulated joint or motorised 'buggy' can be hooked up to the I/O port of a micro via a ribbon cable and operated directly from the keyboard. Building your own buggy (or turtle, as they are often called following their widespread use with computers running Logo) is a little more complicated than it might first appear. The crucial problem is driving the motors and since there are several types of suitable motor (a variety of steppers, DC servos and brushless AC servos) a general article can't go into details. (Although future articles will take up this aspect of robotics — along with others). In general, a motor — like any peripheral — should be chosen with its application in mind, and that means looking at its type, its phase characteristics, torque, speed and supply voltage. These questions become even more complicated when it comes to robot arms, most of which are designed to operate using six motors (one for the gripper and the others for moving the arm with five degrees of freedom).

The dedicated robotics experimenter need not be content with an arm or a leg — at least two American companies have, in the past year, introduced the first of what will doubtless become a flood of 'personal robots'. The Heath Company's Hero I comes as a kit at $1500 (£1600), assembled at $2495, while Nolan K. Bushnell (the man who founded Atari and thus, in effect, the videogames business) is backing Androbot's Topo and BOB ('brains on board') robots at $1195 and $2495 respectively. These robots — resembling R2D2, I'm afraid — come with on-board processors (the 6808 for the Hero and the 8088 for Topo and BOB), arms (which only need to be polar for a mobile robot), and a variety of sensors and speech synthesis devices. Control can be by pre-programming, wire, infrared or radio link.

These robots are capable of mapping rooms, responding to light and sound, fetching and carrying and performing simple household tasks. They can be programmed by a separate computer or using on-board keys or a pendant.

Androbot are promising Topologo and Topoforth software packages on disk to supplement the standard BASIC package. It's an irony that there is precious little in the way of applications software. In the past, we were most concerned about what

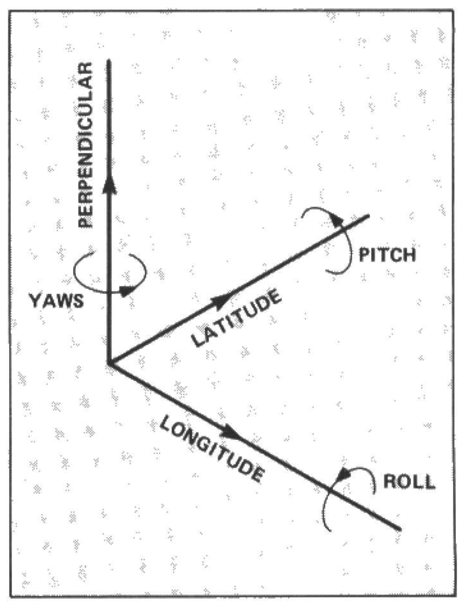
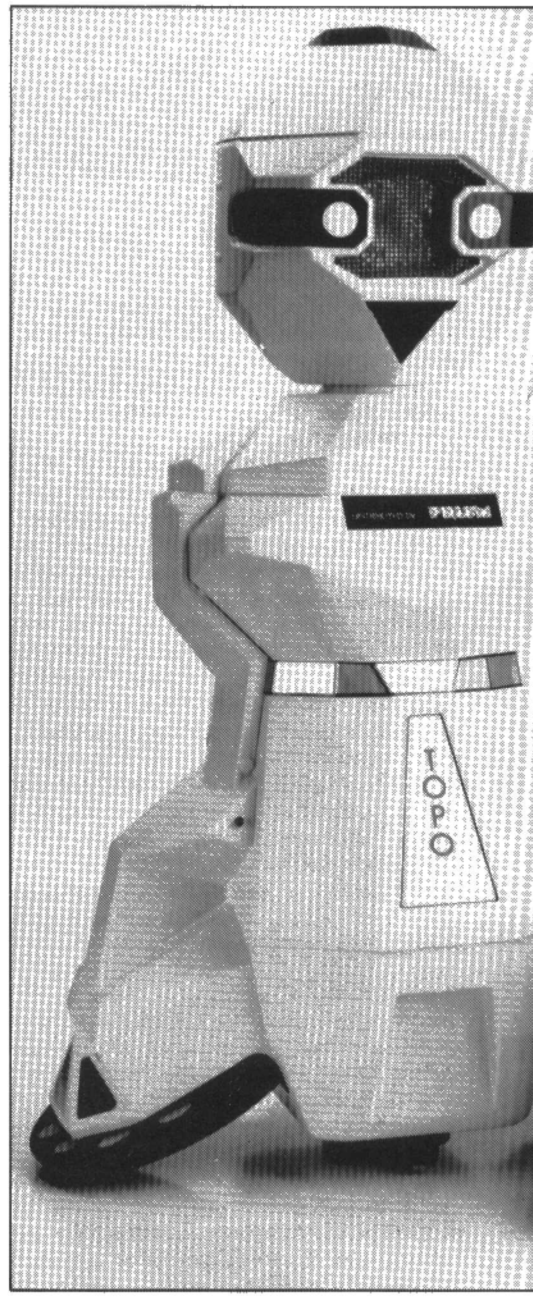


Figure 1. The major axes of a manipulative arm.

robots would do once we let them loose. Now that we have them, it seems that the best their manufacturers can do is teach them to serve drinks at consumer electronics shows. ■

INTRODUCING TOPO

To quote media man on mechanical man, William Woollard said of Topo at its UK launch, 'it is a long way from being a useful domestic robot'. What then, at £1,500, can Topo do that cannot be accomplished by any static computer controlled system?



Topo is a 3 foot high x 2 foot wide robot built in a white ABS plastic body. The robot is controlled via infra red signals from (presently) Apple II and IIe computers, using a version of the FORTH language. And, of course, Topo is all singing, all dancing (it can walk and talk).

Topo moves on two wheels angled at some 30 degrees to the ground. The robot is styled in the manner of those toys which, because of their low centre of gravity, always bounce back to an upright position when pushed over. The DC, encoded motors and gearboxes which drive forward movement (speed, distance, rotation and acceleration) are extremely accurate – to 1%. Speed is 1-50 cm/sec; turn rate is 1-100 degrees/sec; the robot turns within its own width, and has a maximum range of 25 feet from the control transmitter.

Topo does not begin or end a movement at top speed, but with gradual acceleration to speed and gradual deceleration at the end of a movement. This acceleration/deceleration 'ramp' is measured in cm/sec^2 altered in value over a range of 1-255. Curved path movement is also possible using the ARC command (see below). Power is from sealed cell batteries, with a life of 3 hours before recharging.

Topo's speech system is quite versatile. A text-to-speech allophone system is used, converting text into strings of phonetic characters. Topo can control pitch, volume, rate, intonation, and can speak the answers to arithmetic problems (the patient teacher?) or speak a set of words at a certain specified time (Time to get up! perhaps).

Topo is controlled by a FORTH program which uses key words, called Toposoft. At present the only version available is for the Apple II and IIe, but new versions are planned for the BBC micro and other popular home computers. A 6502 assembler already exists in the manual.

Via an RS232C interface an infra red transmitter is connected to the computer. Three banks of transmitter/receivers, capable of penetrating some solid objects, are directed towards the area of movement of the robot. Topo itself has receiver transmitters arranged all round its head, and can therefore remain under control whatever direction it is facing in. The messages received from the computer are decoded and tested. If the message appears garbled or incomplete the robot sends a signal requesting confirmation of the command before acting upon it. Commands may require instant action, or action at some time in the future. Speech and movement commands may be sent simultaneously within a single sequence. A typical command sequence to make Topo turn left and say hello would be:

```
90  LEFT SAY "HELLO"
```

or to make the robot turn in a curved path:

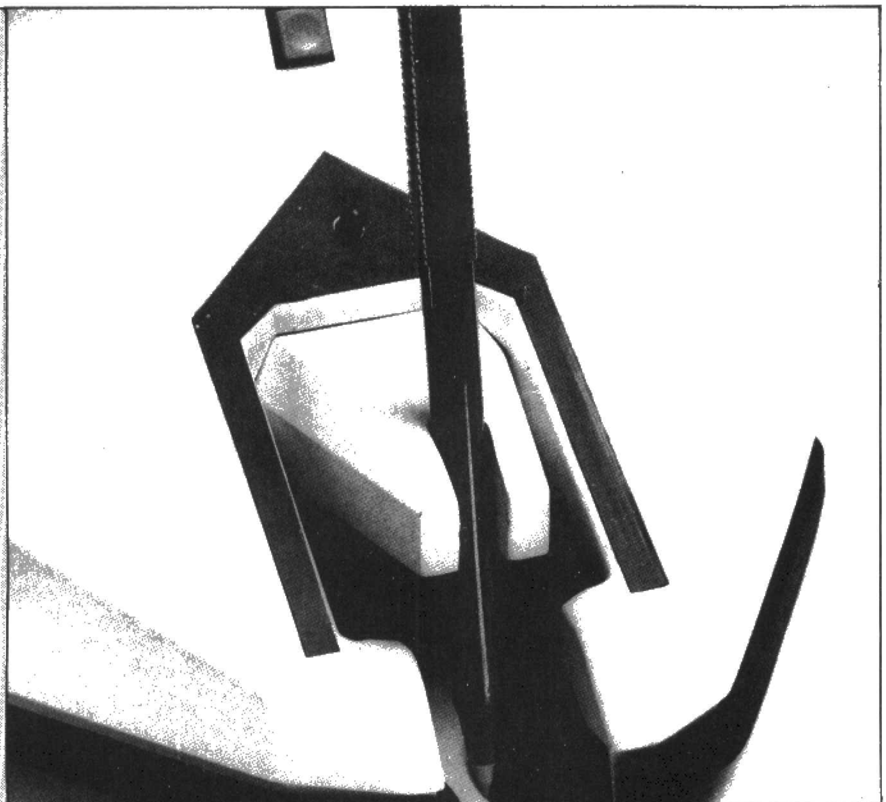
```
90  ARC 100 R
```

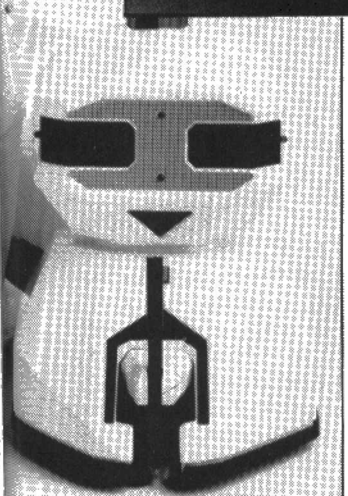
will make Topo perform a 90° right turn while moving forward 100cms.

As well as through keyboard commands, Topo can be controlled by a simple joystick (movements only) or by a button located on its head. By pressing in one (or two) of four directions it is possible to control forward movement (start, stop, right, left). Also available is a teach mode. The robot can be taken through a sequence of movements manually, and these are transmitted to and stored in the computer for later use. Any sequence of instructions keyed into the computer can be nominated as keywords, and the movement repeated by use of the single word or sequence of words in much the same way as the Logo language.

A number of control channels are available for use with future add-ons to

"The uses to which Topo is put will depend largely on its owner's imagination".





Left, Topo and Fred. Fred is a turtle type robot, also controlled by infra red and possessing a speech chip. Above is the moveable arm with which Fred holds his pen.

Topo. A space is available in the side to accept an arm (or 'manipulative growth unit' as the manufacturer terms them) and at the front of the robot for heat, smoke or temperature sensors.

Performance potential

So much for the bare specifications of Topo, what of the performance, and the potential.

Certain deficiences were apparent immediately at the launch of the machine, but these cannot be confirmed until we manage to get hold of one from the UK distributors, Prism. Firstly, it is probable that mobility will be very limited by the type of surface over which the robot will have to move. Topo was designed for maximum efficiency over carpeted floors, and would probably move well over any flat surface from which reasonable purchase could be obtained. At its first day out at the London Hippodrome, however, Topo performed badly on a polished dance floor, failing to rise over a low ledge in the floor (no more than half an inch) and wrecking his/her(?) dance routine.

Topo would probably have some difficulty negotiating the rooms, passages and furniture (to say nothing of small children) within the average house. Topo barely squeezed through the doors of the Hippodrome and flattened one journalist's foot (mine) so breaking Asimov's first law of robotics (quoted in the Topo manual, 'A robot may not injure a human being, or, through inaction, allow a human being to come to harm'). So much for Asimov.

Applications

Topo's superiority over the more common intelligent system, be it speech synthesiser, teaching tool, sensor, timer or alarm, can only be obtained through its mobility. This is probably not yet refined or versatile enough to replace a stationary system: hence the words of William Woollard. The manufacturers of Topo clearly understand this, and are marketing the machine first and foremost as a pioneer of the robotics age, and as a companion.

The first 500 purchasers of Topo will receive a plaque signed by the inventor, Nolan Bushnell of Atari fame, and the robot will be numbered and registered in their name to commemorate the owner's 'leadership in the age of androbotics'(!). So, first and foremost is the prestige of owning your own robot.

Secondly, it is perhaps significant that Topo has a very 'friendly' appearance and is the same size as a child. Psychiatrists may face interesting new problems in the future when visited by all those only children brought up with a robot. Topo would be an ideal teaching device, in both language and mathematics, for the very young.

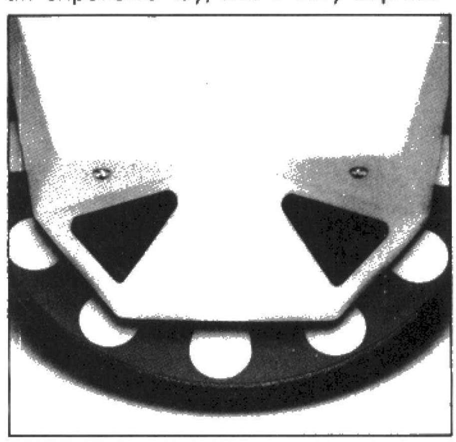
"A pioneer of the robotics age?".

Another obvious applications is in teaching control, but this can be done by robot arms and turtles at less than 10% of the price.

Topo's ability to manipulate objects will probably always restrict its use for simple domestic tasks. A compartment in the robot's chest cavity is intended for carrying objects (the dinner from the kitchen?). Prism, the UK distributors of Topo, claim that it can be used for vacuuming and mowing the lawn. It is difficult to see how such tasks could easily be achieved , particularly the latter.

Finally, the inclusion of sensors will give Topo the capacity to act as a mechanical guard dog, to warn against fire, or to look after the house while the owner is out. Again, however, similar static devices can do the job just as well.

The original question remains partially unanswered. As it stands Topo is an expensive toy, but a very sophisticated one. Toys fulfill an important function in learning, for adults as well as for children. The uses to which the robot is put will depend largely on its owner, and there is clearly potential for serious use. Robotics is a science in its infancy, and Topo may represent the first of a new generation of tools within the home. Equally it and its companions may become white elephants. Until mechanics catches up with electronics in terms of price as well as performance we will not know the answer to this question for Topo or any other robot of its type. The arrival of Bob, a more sophisticated sibling of Topo from the same manufacturer, may provide more clues. bob has an onboard computer and more sophisticated mechanics, and is soon to be launched in the States by Androbot.



Topo moves on two wheels angled at 30° to the ground.

UK robotics research

Peter Matthews surveys the current UK robotics research activity and has some hints for anyone seeking help with the development of a Robotics project.

The academic world is as interested (and maybe even more) in robots and robotic techniques as industry. The research which often reflects the leading edge of technology is sometimes so far advanced that it does not have a relevance for everyday industrial problems. There is however an increasing dialogue between the academic and the businessman. Much of this initiative is the responsibility of the Science and Engineering Research Council (SERC) who administer the funds and resources of Universities, Colleges and Polytechnics. They have by a number of schemes tried to draw the academic and the manufacturing world together on the subject of robots. The result is that there are centres of excellence in robot research all over in the country comparable to anywhere in the world but, although a dialogue is developing, it is not enough.

In 1980 SERC launched a major Industrial Robotics Research programme required for the second generation robot. The SERC definition as "the application and further development of inherently flexible (programmable) devices for improved manufacture and distribution in industry". The looseness of such a definition allowed a wide range of research and development and the academic world has taken advantage of the opportunity. The table opposite of institutions and their research shows their interests.

There has always been an obstacle between the academic and the commercial attitude to product development. The businessman has been known to grumble that timetables, cost effective engineering and serviceability of the product are meaningless to the pedagogue. "He is trying to turn our project into a Nobel Prize" is something that was overheard in one factory. On the other hand the university researchers have been heard to say that they were expected to act as junior production staff. For this reason SERC has put the emphasis on the co-operation of the two sectors using jointly developed objectives. Then using these objectives and the commitment of at least one full time engineer who has the ability to define and monitor milestones, the robotic front stands a far better chance of success than most academic/industrial collaborations.

Under this scheme, over 30 big projects are being carried out in 20 universities costing £3 million. This is the officially funded end of robotic research but this does not describe or define the lesser but just as interesting robotics projects which are going on in polytechnics, colleges and even schools throughout the country. Although the major projects are of national interest I regard the smaller undertakings as being more interesting. Certainly these smaller efforts are far more fun. Predictably these 'fun robotics' are based in the junior and educational range and, of course, the hobbyist. Probably the greatest amount of unsubsidised midnight oil to be burnt is for the 'Micromouse' competition.

This is an event whose presiding genius is Doctor John Billingsley of Portsmouth Polytechnic. His infectious enthusiasm has taken the contest from small beginnings to an international event. The last to be held was in Madrid. The contest takes the form of a maze that the tiny robot has to find its way through to the centre. The complexities of the robot finding its way through, sensing the dead ends or corners and if it wants to stand a chance, remembering where it has been, attracts individuals and teams of the highest calibre. Apprentices from large companies, computer club teams and many others spend much time and trouble to produce beautifully made devices which have become cleverer and cleverer. So much so that in John Billingsleys opinion the concept of the 'race' will have to change. The Micromouse device 'Thumper' not only found its way to the centre in seconds but also announced its progress by voice synthesis as it went. Now John thinks that they will have to redefine the contest. The maze 'race' will be maintained for schools teams and others who want to continue this amusing and interesting event. For those who want a greater technological challenge however, the rules of Robot PING PONG were published in the computer press some time ago. The difficulties of tracking and hitting a ball at speeds almost as fast as a human can move are incredible. The rules which limit the areas of ball movement however make the task easier for participants, although not as easy as the maze race now appears to be to those who have not tried to make their own robot. Not that it seemed that easy or even possible in 1980 when Micromouse started. Maybe we will find the table tennis champions being given a run for their money by a robot in much the same way as the chess masters being given a hard time by computer driven opponents. Whatever happens there will soon be a lot of young (and not so young) enthusiasts building Ping Pong robots. The thing that this will do is extend the knowledge and 'know how' of many in light robotics, much of which will undoubtedly be 'spun off' into more practical applications.

"he's turning our project into a Nobel prize" was something overheard in one factory.

Designing a robot...

... where do you start? Well what's the problem ... do you want to apply robots to your production line or do you want to design a robotics device?

If it is a production problem the place to start without doubt is the PERA. Under the avuncular but enthusiastic leadership of Professor Higginbotham PERA has become one of the foremost centres of robot application in Europe. A

simple request will give a guided tour to their laboratories and workshops showing a whole range of robots and machine tools. They will also give advice and guidance to how to get grants, loans or even free consultancy for the applications of robots to production lines and tasks. The development of a robot or robotic device on the other hand using a research team from one or other of the electronic or mechanical engineering departments of any one of the many universities or colleges requires a different approach. To establish an RGD project with a university the first step would be to seek advice from Doctor Peter Davey at the Rutherford Appleton Laboratory at Didcot about the SERC robotics initiative. He or his staff will give you the necessary advice to take the first steps for collaborative research.

All these are ways to develop major projects and initiatives in robotic development and the grants and loans which come from the Department of Industry will help them along the way. Many readers however will want to develop a system or device which is less ambitious. There are in fact only a few forms of robot that can be involved. There is of course Ⓐ a handling device such as an anthropromorphic or human type of arm or the simpler telescopic form of arm, Ⓑ a mobile robot which could range from the unsophisticated 'turtle' type of device which can only be programmed to repeat a pattern on a floor, to the sophisticated type of Star Wars device which walks and sees where it is going, avoids obstacles, and is able to recognise voice commands and able to respond to them in various ways maybe even by synthesising speech to answer back.

"... throw out a challenge to our readers to design a robot..."

A positional device to repeat better such as the lathe or even a plotter which although not usually regarded as robots do satisfy the criteria which is 'a mechanical device which is capable of being programmed to adapt to the users changing needs or requirements'.

Any robot has mechanical, electronic and programming software aspects to it and the putting together of these in a vastly different combination of ways that lead to the fascination of robot design and application. Most of the component parts of a simple robot in any three classes is already developed and on the market. They consist in general of:-

Mechanics – Gears, drives such as belts or chains, bearings and even some mechanical frameworks such as lego, or mechano although generally robot makers make up their own frameworks with wheels or tracks for driving mobile robots.

Electronics – Driver boards for both servo mechanisms or stepper motors, computer boards for programming the movement of the mechanisms to move either wheels or joints, potentiometers and shaft encoders to measure the position of the wheel or joint and feed the information back to the processing board. Sensors of the tactile or proximity type to detect when the mobile or arm has bumped into something. Vision sensors using photodiodes, cameras or even charge coupled devices, alternatively a sonic or infra red sensor to 'see' obstacles.

Voice synthesis and recognition devices – to give the thing the ability to listen and reply to its master. Last and not least because a power weight ratio is one of the central problems in mobile robots and even to some extent in handling devices – a power source, either transformed from the mains or direct from a battery.

Programs and Interfaces – The easiest thing to put together in a robot is generally the electronics modules. The most difficult part of those electronics is how those modules fit together through interfaces to each other. Lastly a robot, either simple or complex, can be made or marred by the program that runs it. There will be endless discussions about the advantages or otherwise Basic, Machine Code, Forth or other languages for control and how they can be employed to drive operate and flash the lights and eyes of your robot.

These then are the tools of the robot designers trade; let's see how they go together to make devices for fun, education and even profit. Over the next few months we will try to establish a few robot projects that can be carried out by you so that you can better understand the principle and practices of computerised controls, cybernetics and astonishment of your less technical friends.

To do this we are going to go to several institutions mainly of the academic kind to see what they have done to and give you a few specifications, maybe even plans and specified components. We may provide some software but we will expect some constructive suggestions from you as to how you can make the little beast a little more user friendly. We would also like to hear from you as a company, college, polytechnic or individual about the project you have or will develop so that we can tell a world agog to know. We also intend to throw out a challenge to our readers to design a specific robot for a specific purpose and see how you react to the challenge particularly if there is a worthwhile prize at the end of it. Let's see then what next month will bring!!

MAJOR ROBOTIC R & D PROJECTS

Academic Institutions	Co-operating Companies
Aberystwyth	British Robotic Systems
Bath	Waller Crosswaller
Birmingham	Unimation
Cranfield	Remick/Parkinson Cowan
Cambridge	—
Durham	Lyle & Scott
Edinburgh	GEC Hirst
Gwent College of Higher Education	—
Hull	GEC Marconi/Corah
Imperial College	—
Lancaster Polytechnic	LK Tools
Liverpool Polytechnic	—
Loughborough	Martonanic/British Rail/Coates Patons
Newcastle	Austin Pickersgill
Nottingham	—
Open University	—
Oxford	BL Technology
PERA	—
Portsmouth Polytechnic	Turnwright Controls
Queen Mary College	Micro Consultants
Queens University Belfast	Short Bros
Royal Holloway College	United Biscuits
Rutherford Appleton Laboratories	Computer Recognition Systems
Salford	Fairy Automation
Surrey University	Hall Automation
Sussex University	EMI
UMIST	British Robotic Systems
University College	—
Warwick	Lansing Bagnall

Academic institutions involved in robotics research together with any co-operating commercial company.